Computer-Assisted Musical Instrument Tutoring with Targeted Exercises

by

Graham Keith Percival

B.A., Simon Fraser University, 2003

B.Mus., University of Victoria, 2006

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

Master of Arts

in Interdisciplinary Studies (Computer Science and Music)

University of Victoria

Computer-Assisted Musical Instrument Tutoring with Targeted Exercises

by

Graham Keith Percival

B.A., Simon Fraser University, 2003

B.Mus., University of Victoria, 2006

Supervisory Committee

Dr. G. Tzanetakis, Co-Supervisor (Departments of Computer Science and Music)

Dr. A. Schloss, Co-Supervisor (Departments of Computer Science and Music)

Dr. J. Goldman, Member (Department of Music)

Supervisory Committee

Dr. G. Tzanetakis, Co-Supervisor (Departments of Computer Science and Music)

Dr. A. Schloss, Co-Supervisor (Departments of Computer Science and Music)

Dr. J. Goldman, Member (Department of Music)

Abstract

Learning to play a musical instrument is a daunting task. Musicians must execute unusual physical movements within very tight tolerances, and must continually adjust their bodies in response to auditory feedback. However, most beginners lack the ability to accurately evaluate their own sound. We therefore turn to computers to analyze the student's performance. By extracting certain information from the audio, computers can provide accurate and objective feedback to students.

This thesis lays out some general principles for such projects, and introduces tools to help practicing rhythms and violin intonation. There are three distinct portions to this research: automatic exercise creation, audio analysis, and visualization of errors. Exercises were created with Constraint Satisfaction Programming, audio analysis was performed with amplitude and pitch detection, and errors were displayed with a novel graphical interface. This led to the creation of MEAWS, an open-source program for music students.

Table of Contents

List of Tables

List of Figures

Symbols and Definitions

$\wedge$	Logical AND	$\vee$	Logical OR
$\implies$	Logical implication	$\iff$	Logical equivalence
$\forall$	For all	$\in$	Set membership
$\sum$	Sum	$\bar{X}$	Mean (average) of list X
$\frac{d}{dx}$	Differentiate with respect to x	$x \leftarrow y$	Assign to x the value of y

Certain musical terms have specific meanings in Computer Music which may be counterintuitive to musicians.

Performance: the act of producing sound. To a musician, this term implies a certain level of skill or the existence of an audience (i.e. "a concert performance"); in this thesis, the term applies to *any* sound production (i.e. "the drunken bassoonist grabbed a violin and performed a Ševčík exercise").

Transcription: the process of determining the notes and rhythms in a performance. The production of sheet music ("engraving") or printing physical output is not required.

Preface

As is always the case with children and parents, I did not realize how lucky I was to have the parents I did while I was a child.

Such a statement could go in many directions, but in this case I am referring to their help while I learned cello. Mom tuned my instrument before I practiced (with Dad assisting when the pegs got stuck and required force), and monitored my daily practice for the first 6 years. She corrected my intonation, played the piano, and generally ensured that I was improving during my daily practice.

I didn't appreciate how useful this was until twenty years later, when I started teaching cello myself. I saw my students once a week. I tuned their instruments – sometimes with strings that were a major third out of tune – and carefully listened to their scales, pointing out glaring intonation mistakes. And six days a week they practiced on instruments which were horribly out of tune and without anybody who could check their intonation. In other words, they spent six days out of seven reinforcing bad habits.

This is not to blame the students or their families – unless a student has perfect pitch, it takes years to develop the ear training necessary to judge string instruments like the cello. At the time, I recognized the problem, but considered it insolvable.

The second inspiration for this thesis came a few years ago at a summer music camp for amateur musicians. I was playing violin in a string quartet, and the cellist – as it happened, a professor of Computer Science (although not in my area) – was struggling to play his solo. The rhythm switched between duple and triple time, and it always threw him off.

This was hardly the first (or the last!) time that I'd played with an amateur musician who had difficulty with something that I found trivial, but this particular case left a lasting impression on me. In part because my first instrument was the cello, and this was one of the first times I had played something other than cello in a string quartet. But mostly because I knew that he was neither an idiot nor lazy – he spent far more time practicing than I did. He listened to recordings, he tried clapping the rhythms, he tried mouthing two- and three-syllable words as he played... in the

end, the quartet simply followed whatever rhythm he played at that point, and fixed the tempo a few bars later. However, this problem stayed in the back of my mind – even intelligent, hard-working amateur musicians could have enormous difficulty evaluating their performance.

The third inspiration came one month after beginning this degree. I was attending ISMIR 2006 (International Conference on Music Information Retrieval), and one of the speakers was describing how he became interested in music transcription. A few years ago, his neighbor's six-year old son began learning violin, and the speaker was horrified at the sounds that he heard – "they were not music!". So he asked himself, as somebody skilled in digital signal processing, if there was anything he could do to improve beginning violin students.

I was quite impressed by his courage – knowing nothing about playing the violin or teaching music, here was somebody working to improve the education of millions of violin students with non-musical families. With all my expertise in playing, teaching, and conducting string instruments, how could I do anything less?

Acknowledgements

I wish to thank my supervisors George Tzanetakis and Andy Schloss for giving me the freedom to switch from one project to another. A pattern arose eventually – Computer-Assisted Musical Instrument Tutoring – but this was not apparent to anybody at the time. I am also eternally grateful that they accepted me as a student in Computer Science as well as Music – in other words, that when I said "I am one of the main GNU/LilyPond developers" they thought this meant that I was a competent programmer. (I didn't, and wasn't; my role in LilyPond was writing documentation and organizing bug reports.)

I would like to thank my lab-mates Luis Gustavo Martins and Mathieu Lagrange. Gustavo taught me how to use Marsyas and basic knowledge such as C++'s `const` keyword and how to perform autocorrelation. Mathieu added an improved pitch-detection algorithm to Marsyas and discussed many aspects of Computer-Assisted Musical Instrument Tutoring with me.

I received financial support in my second year from a UVic Interdisciplinary Fellowship. This provided a much-needed break from being a Teaching Assistant, allowing me to actually begin writing this thesis.

Finally, I would like to thank my brother and unofficial supervisor, Colin Percival. His faith in my ability to do Computer Science re-introduced me to this field.

Chapter 1

Introduction

This research focuses on the practical aspects of learning a musical instrument and how they may be improved with Computer-Assisted Musical Instrument Tutoring (CAMIT) software. Beginning music students face four challenges: learning musical theory, learning the theory of playing their instrument, developing the muscle control to play their instrument, and developing the ability to listen and judge the correctness of their sound. The first two challenges are theoretical, while the latter two challenges are practical.

The focus on practical challenges should not be taken as suggesting that theoretical challenges are trivial, either for students or teachers. Learning how to read sheet music or memorizing hundreds of folk tunes is certainly difficult, and learning how to play stylistically and beautifully is a life-long task. However, the theoretical challenges are relatively well understood. Educational software aimed at teaching students how to read sheet music has been around for decades. Such software is certainly not perfect, but there are many other researchers investigating this area. In addition, teaching the academic side of music is not terribly different from other academic subjects. New advances in computer-aided education teaching math or history can probably be applied to teaching music theory relatively easily.

In contrast, few researchers working on CAMIT software focus on the practical aspects. Playing a musical instrument involves quite unnatural movements, with very little tolerance for deviations while performing them. Most mistakes made by music students are caused, directly or indirectly, from insufficient muscle control – either failing to move their bodies as intended, or concentrating so much on their body movements that they lose track of the music. In such cases, the student does not need a review of the theory; they merely need to practice the exercise again to improve their muscle control.

This thesis is divided into five chapters:

Chapter 1 Introduction explains the motivation and general principles behind this research. It explains the particular difficulties of learning musical instruments, lays out some guidelines for effective future work in this area, and examines related work.

Chapter 2 Creating Exercises shows how exercises for students were created using simple music representations with Constraint Satisfaction Programming.

Chapter 3 Exercise Analysis shows how student performances were transcribed from audio into musical information, and how that musical information was graded.

Chapter 4 Game Design and Visualization explains the design of the rhythm and violin intonation games, and how the result of the analyses were presented to the students.

Chapter 5 Conclusion and Future Work contains some final remarks and plans for future work.

1.1 Difficulties in Learning Musical Instruments

This section examines particular challenges faced when learning a musical instrument.

1.1.1 Athletics and Verification

The practical side of learning how to play a musical instrument is quite different from learning academic subjects like math or history. The difference boils down to two factors: athletics and verification.

Non-musicians are often surprised to hear the terms "athletic" and "musicians" together, but musicians *are* highly-trained athletes. Their training is not primarily directed at raw strength or endurance, but is instead focused on muscle control. Musicians must control parts of their bodies very precisely. If a violinist's finger is 3 millimeters too high or low, we hear an out-of-tune note. If a clarinetist's lip pressure is too loose or too great, we hear a breathy whisper or no sound at all.

To illustrate the difficulty of muscle control, try performing a simple experiment. Place the finger-tips of your left hand on a table. Now shift your hand so that your index finger is touching the table exactly where your ring finger was touching. The new finger should be accurate to within 3 millimeters. Now shift your hand back to its original position – again, your finger should be within 3 millimeters of the position of the previous finger.

Can you perform this exercise quickly? Moving your hand should be completed within a tenth of a second. Can you perform this exercise without looking at your hand? Can you move your right hand smoothly in a line in the air while your left hand performs these fast – yet accurate – movements? This is the kind of muscle control which violin students must learn in their first year. The difficulty is not *knowing* what to do, but simply in *controlling* one's body.

As we perform this experiment, we notice a second difficulty: verification. If we do not look at our left hand, how can we tell if our new finger is in exactly the right position? In this case, our experiment was harder than a real-life violin lesson: when we perform these movements on a violin, the position of our hand determines the pitch. If our finger is in the wrong position, the pitch produced by the instrument will not be correct.

The difficulty of verification now becomes one of hearing. Almost all humans have the ability to distinguish different pitches, but the accuracy varies widely. With the exception of medically "tone deaf" individuals, all humans can recognize that 10,000 Hz is a higher sound than 100 Hz. On the other side of the scale, no human can recognize the difference between 440 Hz and 440.01 Hz; this is far below the just noticeable difference.[1] To be recognized as playing "in tune", the musician's margin of error must be less than the audience's ability to detect pitches.

The previous paragraph was directed at stringed instruments such as the violin, but the problem of verification is true of all instruments. Instruments which have fixed pitch (such as the piano or drums) still have variable rhythm. For example, consider an exercise as simple as playing Figure 1.1. The rhythm is clear: for example, if one plays the initial C for 2 seconds, then the D and E should be 1 second each, and the F should be 4 seconds.

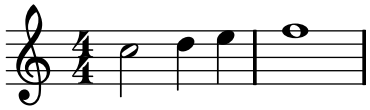


Figure 1.1: A very simple melody.

In practice, nobody plays rhythms so strictly – quite apart from the non-musicality of playing in strict "metronome time", no human can judge whether a note was 0.001 seconds too long or short. However, if a note is 1 second too long, we definitely *will*

[1] Our ability to distinguish pitches varies enormously with the timbre, tonal context [26], and whether we hear the tones simultaneously or successively, but most studies suggest that we cannot perceive pitch differences below 5-6 cents for sequential notes such as played by a beginning violin student [24] 5 cents higher than 440 Hz is 441.273 Hz.

notice a problem. To be recognized as playing the rhythm "correctly", the musician's margin of error must be less than the audience's ability to detect such errors.

The practical difficulty for beginning music students is now clear: they require hours of practice to control their bodies, but they cannot even judge if they are performing their exercises correctly or not.

1.1.2 Description of Daily Practice

To improve our understanding of the challenges faced by music students, we shall describe ther daily home practice. This description is typical for classical European instruments such as violin, oboe, or piano. Some portions of this daily routine may be omitted for certain instruments, but the basic framework is the same.

The practice session begins by playing scales. The purpose of playing scales varies based on instrument; goals include intonation (producing the correct pitches), speed, and good sound quality in all ranges of the instrument (high notes and low notes).

Scales are followed by technical exercises. These are generally quite short; many exercise are between two and ten seconds, but there are a large number of exercises to be played. These exercises are rarely "musical" (aesthetically pleasing); they are the instrumental equivalent of weight training. Many technical exercises involve playing notes very rapidly with accurate intonation, some involve playing notes which are quite distant, and other exercises simply ask the student to play a long note with steady pitch, loudness, and tone quality. Depending on the seriousness of the student and the teacher's approach to music education, these technical exercises may be omitted entirely – very few students enjoy performing these exercises, and many teachers consider fostering an enjoyment of music to be more important than improving a student's ability as quickly as possible. The analogy to weight training is also applicable here: if one simply wishes to play team sports for fun, weight training is not necessary; if one wishes to play competitively, extra physical training is required to improve quickly.

Next a student will play a 'study' or 'étude'. These lie somewhere between technical exercises and normal piece of music: each study is specially composed to stress certain technical skills (playing high notes, producing a certain kind of sound, etc.), but a study *is* musical. Studies are also much longer than technical exercises; most are between one and ten minutes long.

Finally, a student will begin playing pieces of music. Depending on the student's age and ability, there will generally be two to five pieces of music. The teacher will often ask the student to pay particular attention to a few challenging parts, but for

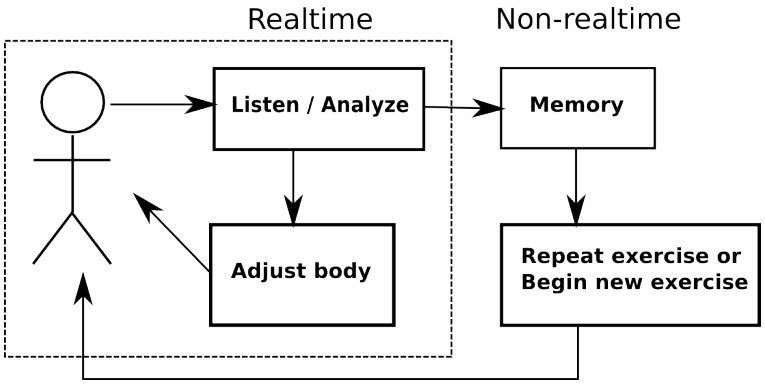


Figure 1.2: Schematic of feedback cycles in daily practice.

the most part the student must do her best to improve her performance by herself.

At every stage of this daily practice, the student should be analyzing her own performance as shown in Figure 1.2. There are two feedback cycles: realtime and non-realtime. In the realtime feedback cycle, the student is performing tiny adjustments to her body in response to the sound – adjusting fingers, lips, air flow, etc. After a student has finished playing, she must decide whether to continue practicing the same material (and if so, what part(s) she should fix next), or whether to move on to new material (another exercise, another piece of music, or cease practicing).

In her previous lesson, the teacher will have pointed out a few mistakes of the student, and directed the student to correct these mistakes. However, the student should not simply correct the mistakes mentioned. Due to constraints on time (and the student's concentration), teachers only mention a small fraction of the mistakes they notice. If a student has fixed the mistakes discussed in the previous lesson, she should look for other mistakes to fix. In addition, the student may have developed new mistakes in the time between lessons. Another problem is that the mistakes discussed in the previous lesson are not always obvious to the student, even once the teacher has discussed the issue.

For example, suppose that the student was warned that she always played a certain note too high. In some cases – particularly with advanced students – the student can fix the mistake herself. However, less experienced students may lack the ability to hear the difference in pitches. The student may honestly believe that she played the note at the correct pitch when in fact it was extremely out of tune. When the teacher is present, she may analyze the student's sound and notify the student of mistakes, but without the trained ears of the teacher, the student is helpless to fix the mistake.

1.2 Goals of CAMIT projects

Before beginning work on any complex system, we should have a clear concept of the goal(s). This section will examine potential goals and discuss which goals will produce the greatest ratio of student help for researcher effort.

1.2.1 How Computers *Could* Help

There are three potential goals of CAMIT projects: enhancing the teacher's lesson with the student(s), enhancing the student's individual practice, and motivating the student. Most projects will pursue two of these goals (i.e. motivation and either enhancing lessons or enhancing individual practice), but in some cases it may be useful to pick a single goal. For example, when dealing with highly-motivated students (either competitive teenagers or adult beginners) the problem of motivation might not arise. Some students may even consider extra "motivation multimedia" (e.g., a talking dinosaur, a bouncing smiley face, a game defending the earth against an alien invasion) insulting, and may prefer a CAMIT project which allows them to disable any extra "motivational multimedia". Conversely, for some students (intelligent, talented, but easily bored) motivation may be the *only* problem that needs addressing. They may be perfectly capable of learning by themselves and correcting their own mistakes, but may lack the willpower to actually sit down and play their instrument every day.

The teacher's lessons

The study of computer-assisted music education is relatively young, but it is unlikely that human teachers will be replaced in the foreseeable future. Teaching humans – especially children – requires a mixture of subject-specific knowledge, communications ability, psychological skills, and creativity. In addition to evaluating a performance and deciding which mistakes to discuss, a teacher needs to evaluate the student's mood and interest level. If a young student is feeling discouraged, it may be desirable to compliment the student's performance regardless of any problems, and suggest a new piece of music to work on. Conversely, a perfectionist student may prefer to continue working on the same piece of music much longer than other students. Until we have developed artificial intelligence which can evaluate a child's mood based on her body language, we should be thinking of ways to enhance the lessons of human teachers, not replace them.

Using technology to enhance music lessons is nothing new. Some teachers use mirrors so that they can easily monitor the student's movements from multiple angles (or use the same mirrors to demonstrate their own movements to the student). Many

teachers use recording devices (cassette tapes, minidiscs, or computers) to record their students and then play the student's performance with the teacher's commentary.

We can easily apply these same examples to educational multimedia. Instead of setting up a single mirror so that a student in the same room can view the teacher's demonstration, we could set up multiple video cameras so that the demonstration can be viewed by many students in various geographic locations and potentially tens of years in the future. We could further improve our archiving of performances by using body sensors of the kind used by some dancers to produce computer animation [38]. The resulting data can produce a computer animation which may be viewed at any position, angle, or speed.

Individual practice

The vast majority of a student's time with their instrument is spent on individual practice. Individual practice is less effective than lessons with a teacher, but due to economics and practicality, most students have one lesson per week. The effectiveness of individual practice is therefore absolutely vital to a student's progress. Effective individual practice is particularly difficult for young children; for this reason, several approaches to music education (notably the Suzuki method) stress parental involvement in lessons and supervision of home practice.

There are a number of existing technologies to improve the effectiveness of individual practice. Two early inventions were the tuning fork (a metal device which vibrates at a fixed, known frequency) and the metronome (a device which plays a sound at regular intervals; generally between 30–240 beats per minute). In the late 20^{th} century, electronic tuners replaced tuning forks – students could see an electronic device's estimation of their current pitch, displayed along with pitches of nearby notes.

With multimedia computer programs, we can significantly improve on these old technologies. Instead of comparing pitches (audible pitch vs. pitch of nearby real notes) at individual moments in time, we could compare the pitches in an entire piece. A student could perform an exercise, and a computer could compare the student's pitches with the expected pitches. The computer could then highlight the three worst notes and inform the student, who would then perform the exercise again.

Motivation

Humans are immensely lazy creatures. We are also very creative; this is an unfortunate combination. We are extremely skilled at finding excuses to avoid anything that resembles work, such as practicing musical exercises, fixing mistakes in said exercises, or even taking our instruments out of their cases.

Some people may consider student motivation to be outside the specific area of computer-assisted music education; motivation is a general problem in education and "edutainment" computer programs. There are certainly many problems we can research without regard for motivating students – multimedia analysis, creating multimedia feedback for students, etc. However, the single most useful factor in any practical multimedia system for students is motivation. If students could be motivated to play their assigned musical exercises every day, that would prove more useful than even the fanciest multimedia feedback systems.

There are many ways to motivate students; first we must identify our target audience. For young children, it might be appropriate to display brightly-colored stars. Older children may enjoy the notion of "gaining experience" and "going up levels" – possibly within the framework of a game where the user is attempting to save a princess or defeat an evil wizard. If the target is adult males, then perhaps the ability to compare their scores competitively with other users on the Internet would motivate them to practice their scales.

1.2.2 How Computers *Should* Help

Although there are a few ways that multimedia tools may enrich the lessons of a teacher, computer research in this area is likely to be less effective than work in the other two areas. First, the time a student spends with a teacher is far exceeded by the time spent without one. Many teachers suggest that their students spend an equal amount of time in daily practice as they do in lessons (i.e. a one-hour weekly lesson would result in one hour of daily practice). Depending on the seriousness of the student, the practice time may be double or even quadruple this amount (i.e. a one-hour weekly lesson would result in four hours of daily practice). Second, music teachers are already investigating this area. As new technology becomes available, some music teachers incorporate it in their lessons. Such "teacher-driven" innovation will likely be customized to that teacher's specific teaching style, and therefore will be used much more than an externally-created CAMIT project.

The question of motivation is a current area of research in departments of Education – and in the design of computer games. It is to the latter area that we wish turn turn: millions of children and adults spend hours each day playing computer games. Many game-players even pay a monthly fee to play online games. If we could design a computer-assisted music education program that was *half* as addictive as the leading Massively-Multiplayer Online Role-Playing Game (MMORPG), this problem would be solved.

In contrast, there is relatively little attention focused on individual practice. Music students rarely use technological aids – they may use a tuning fork or electronic tuner at the beginning of a session to tune their instrument, and they may occasionally use a metronome, but those are infrequent. Many music students find it difficult to accurately judge their own performance. This is quite problematic because, as the famous phrase goes, "practice makes permanent". If a well-meaning child spends 90% of her time practicing mistakes, her teacher will have a hard time correcting those mistakes.

There are a few reasons for the difficulty of self-analysis. To some extent this is a physical difficulty – the location of the musician's ears compared to the audience's ears. In addition, controlling an unfamiliar musical instrument requires a vast amount of concentration; beginners simply have no cognitive power left to listen to their sound, let alone critically analyze their performance. With more experience, instrumental control becomes subconscious (much like learning how to walk or drive a car), but this process takes years. We can avoid the physical problems (location of ears, and concentrating on physically controlling the instrument) by recording a performance and listening to the result, but this would double the time spent on individual practice. Most students are not willing to do this on a regular basis.

However, the primary difficulty in self-analysis lies in the student's inexperience. Music students lack the ear training that professional musicians have; a student may not notice subtle errors in the sound, or even if they are aware of the presence of an error, they may be unable to pinpoint the source.

1.3 Enhancing Individual Practice

In Section 1.2.2, individual practice was identified as the most important area for computer enrichment, and in Section 1.1.2 the activities that take place during individual practice were examined. Given the discussion so far, there are two areas that would benefit the most from multimedia computer programs: computer analysis to provide objective self-testing tools, and motivation to practice technical exercises.

We suggest focusing on technical exercises as an ideal candidate for both tasks. Since each exercise has a specific purpose, the computer analysis is much easier. We can develop algorithms to analyze sound for the particular features that are being tested in each particular exercise. Analyzing a five-second audio file for one or two features (such as pitch stability or a gradual increase in amplitude) is much easier than analyzing a five-minute piece of music to find all musical mistakes. This also simplifies the task of giving feedback, since the student is expecting only one or two

metrics about their performance.

Computer analysis of technical exercises is also easier to verify. If we give the recorded audio of three students playing a piece of music to three music teachers, we will receive three (or more!) different opinions about which problems the students should work on. In these cases, it would be very difficult to demonstrate that a computer analysis of the performance was accurate. On the other hand, if we give recordings of a technical exercise, and ask the teachers to rank the students' intonation ability, the teachers will probably rank the students in the same manner. In this case, we have a clear goal for our computer analysis.

1.3.1 Motivating exercises: Educational games

In addition to having a clear focus for each computer analysis tool, technical exercises have the greatest need of the extra motivation and 'fun' that multimedia tools can provide. Here we shall examine three successful educational games which provide inspiration for musical edutainment projects.

Project LRNJ: memorizing Japanese kana

LRNJ [20] is a simple downloadable game which teaches users the Japanese alphabets (*hiragana, katakana,* and *kanji*), shown in Figure 1.3. The game is basically a simple flash-card memorization tool: it displays a character and asks the user to type its English equivalent (for *hiragana* and *katakana*) or the meaning (for *kanji*). If the user guesses incorrectly, the game displays the correct answer, and asks another question. However, this game 'dresses up' these flashcard questions in the guise of a 1980-style role-playing game. Slime monsters have kidnapped the princess of the kingdom, and the user (playing a poor farmer called Jenk) must rescue the princess. Jenk must visit towns, talk to villagers, and fight slimes. But instead of simply attacking a slime and inflicting damage (as is customary in RPGs), the user is presented with a Japanese character. If the user correctly identifies the character, then the slime is damaged or killed; if the user makes a mistake, the slime is healed.

By wrapping a boring task (memorizing over a hundred characters, in addition to nearly two thousand *kanji*) in the guise of a nostalgic computer game, the task became much less tedious. I had previously attempted to learn *hiragana* and *katakana*, but having very little patience for pure memorization, had given up after only five minutes. When I discovered LRNJ, I played the game continually for six hours. It should be emphasized that the true value of LRNJ lies not in any sophisticated interactive tutoring system; it was the extra motivation.

Figure 1.3: Screenshot of LRNJ testing *kanji* recognition.

LOOM: a musical adventure computer game

LOOM [44] was an innovative adventure computer game created by LucasArts, published in the early 1990s and shown in Figure 1.4. In this adventure game, the main character is a Weaver, a person who can cast spells by performing short melodies. Some melodies may be read in books, but others must be learned from the environment. For example, to learn the "see in the dark" spell, the user must listen to the song that an owl sings, then replicate the notes. Certain spells must be cast in order to progress through the game – for example, the player cannot navigate a maze in the dark without casting the "see in the dark" spell.

At the time, microphones and sound cards were very rare. Melodies were performed by typing note letters on the keyboard, such as "`C G E G F E D C`". The musical training here was simply interval recognition ear training, but by presenting this task as an adventure game, it became more amusing.

Guitar Hero and Rock Band: graded performance on fake instruments

The Guitar Hero series of games [43], as well as their predecessors Frequency and Amplitude, are musical games for gaming consoles. The basic premise is similar to karaoke: the user must 'perform' certain preset songs. In the earlier games, this performance was created with the standard console game controller; with the Guitar Hero games, the user plays on specially-made mock guitars.

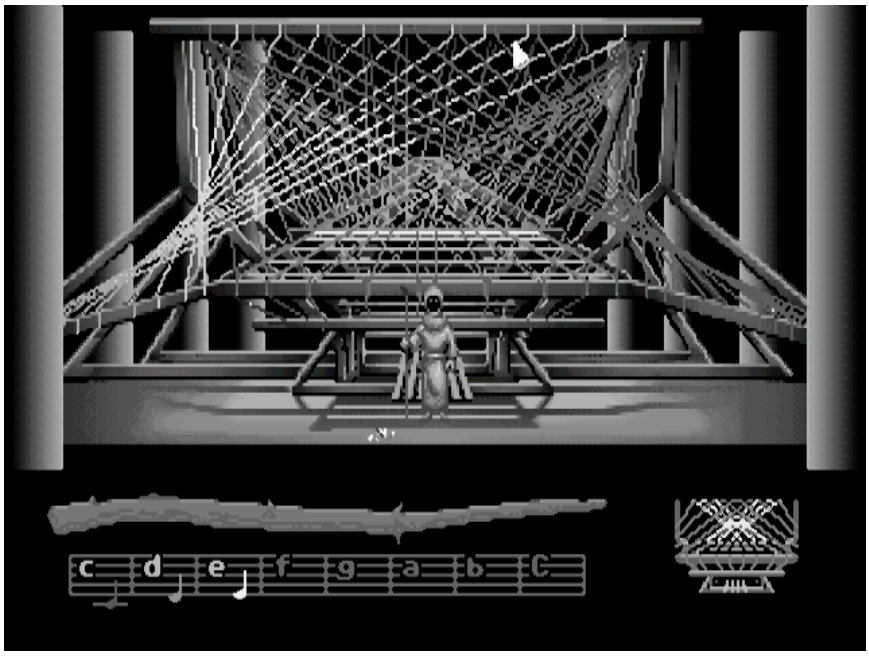


Figure 1.4: Screenshot of LOOM; only three notes (CDE) are available. As the game progresses, more notes are unlocked.

The game Rock Band [45], released in late 2007 and shown in Figure 1.5, carries this idea one step further. In addition to a mock guitar interface, gamers may use a mock bass guitar, electronic drum pads, and a microphone for vocalists. From the standpoint of music education, the addition of the drum pads and microphone are particularly interesting.

Skills learned while practicing with the mock guitar do not translate to real guitars. The guitar controllers do not have any strings; users must press buttons (arranged in frets) which are as wide as the instrument's neck with their left hand while performing a sweeping motion with their right hand. In the early difficulty levels, users press only one button at once, but later levels require pressing multiple buttons. However, since there are no distinct strings, their fingers may line up completely unlike on a real guitar.

In contrast, drum pads and the microphone offer actual musical training. Electronic drum pads are popular in real rock bands, so playing drums in Rock Band develops skills which may be applied directly to a real rock band. Similarly, training to sing the correct pitch with the microphone in Rock Band could aid in singing away from the game.

Guitar Hero and Rock Band use an unusual form of music notation: a vertically scrolling display. The actions to perform at each moment in time are displayed on

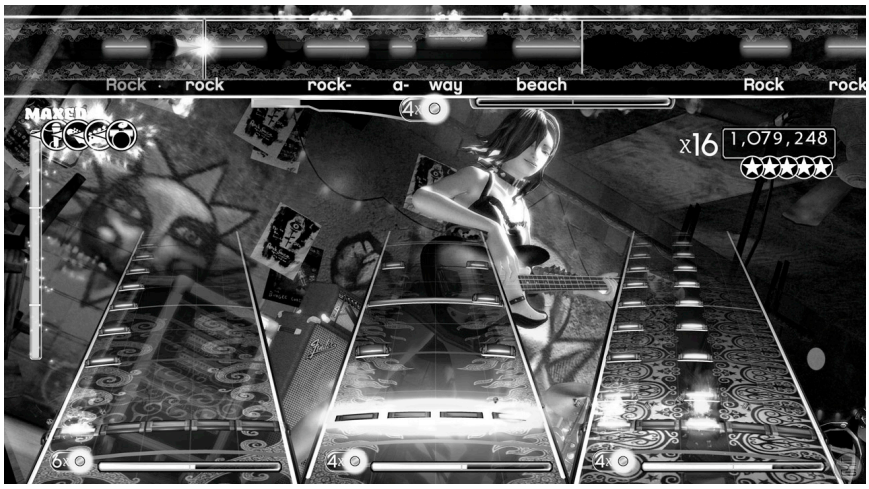


Figure 1.5: Screenshot of Rock Band. The top portion shows the vocal pitches and words, while the bottom portion shows the bass guitar, drums, and lead guitar notes from left to right. For the bottom portion of the display, the notes scroll vertically from top to bottom. Playing a note correctly causes it to flash.

horizontal lines; the user can see the upcoming actions for approximately two seconds. Given that few rock musicians can read standard sheet music, inventing a new form of notation was desirable. Although it is not as compact as Western sheet music and does not adapt well to a fixed form (such as paper), it is quite appropriate for a video game.

Both games feature less than one hundred pre-set songs, with an option to download extra songs on some platforms. Rock Band has different difficulty levels for different instruments; for example, the band may play one song with the singer on Hard, the drummer on Easy, and the lead guitar on Medium. The ability to integrate multiple difficulty levels while playing the same song is vital for such a party game: a random collection of friends will likely include complete novices as well as experts.

1.3.2 A Game for Classical Instruments

Guitar Hero / Rock Band and LOOM both offer a clear vision of education games for musical technical exercises. Instead of 'performing' music with a replica guitar, why not perform it on a real acoustic guitar, oboe, or violin? Instead of typing "`C G F D`" to cast a magic spell, we should have the user perform the melody on their instrument.

We could easily add some practice / repetition to this task: suppose that the hero in our adventure game must climb over a wall. She knows the "create ladder" spell, so she performs the exercise (playing a steady pitch with constantly-rising amplitude)

on her violin. The computer analyzes the sound and gives the user a score of 70%. Instead of giving this number to the user as feedback, the game draws a ladder growing from the ground – but stopping before it reaches the top of the wall. The user must then perform the melody again – perhaps ten or fifteen times in a row – until she receives an acceptable score and the ladder reaches the top of the wall. Depending on the test, the user may even be required to score above 70% three times in a row.

Designing the human-computer interface for such a game is a non-trivial task. We do not want users to have their hands on the keyboard (as is the case with LRNJ, LOOM, and many other edutainment games). Users should be playing their instrument as much as possible. There are three options for input: we could use a controller which does not require the user to release their instrument, we could control the game via the sound of the instrument itself, or the game could have distinct keyboard-controlling and instrument-controlling modes (hopefully with few changes between the modes).

The advantage of using a controller such as a typical console game controller is that the interface is easily recognized by our target audience. The buttons on the controller (e.g., up, left, A, select) are easily mapped to the actions in the game. The disadvantage of such a controller is that the user is not playing her instrument. A one-handed controller minimizes this problem – the user could remove a single hand from the instrument while still holding it, provided that the actions within the game were short – but the problem remains. As an alternative, foot pedal(s) may also be used – but this would require students to purchase such controllers.

We could also control the game via the sound of the instrument. By performing pitch detection (and possibly the whole music transcription chain, involving onset detection and note segmentation), we could control actions within the game. The advantage is clear: the user is always playing her instrument, either by playing exercises to advance through the game, or simply providing audio commands to the game. The disadvantage is that the game-controlling sounds are not very intuitive. For example, consider the simple problem of moving a character within the game. A high pitch could move the character up and a low pitch could move the character down, but there is no intuitive mapping between particular sounds and left/right movement.

In some genres of games, an unintuitive mapping between controls and game actions may not be a problem. Many fighting games deliberately include "special moves" which require complex patterns of key presses. For example, a character in a

martial arts-themed game might throw a lightning bolt when the user hits left-left-A-down-B-down-up. Why not use a violin sound to produce such moves? Normal notes in first position could correspond to simple punches or dodges, while playing chords in higher positions could create special moves. An out-of-tune note (or chord) would simply not be recognized by the game. To force the user to use the whole range of their instrument, we would limit each note to only one use – playing the open E string would launch a simple kick, but if the player wishes to kick again, she must play the same note an octave higher.

A third option would be to minimize user interaction in the game. One model for this is the genre of "visual novels", popular in Japan but very infrequent in North America. After starting a game (perhaps involving typing a character name and selecting a difficulty level), a story unfolds. At certain points, the user would be prompted to perform an exercise. Based on the results of the exercise, the story progresses in a different manner. For example, if the story revolves around a high school, when the main character attempts to invite a girl to dance the user is prompted to play a two-octave F♯ major scale on his violin. If the scale is badly out of tune, the girl in the game slaps the main character. If the scale is in tune, the girl agrees to accompany the character. If the scale is on the boundary, the girl may reply "I'm not sure... ask me again!" and the user is given another chance to perform the exercise.

Although some of these ideas may seem silly (martial arts fighting with a violin?), they are motivated by serious concerns. Many children spend dozens or hundreds of hours practicing their game skills with video game controllers, keyboards, and mice. Skilled gamers can perform incredible feats of dexterity with their hands. If we could tap into this motivation and direct it towards musical instruments, students could benefit immensely. If we can tie a reward (beating all challengers with a triple-volcano-punch) to the player's ability to shift between first and fifth positions on the violin, some students will practice their shifts like never before.

As we saw in the discussion about the LRNJ game, edutainment software does not necessarily need to include sophisticated analysis and teaching algorithms. Once again: if students could be motivated to play their assigned musical exercises every day, that would prove more useful than even the fanciest multimedia feedback systems.

1.3.3 Design Goals for Enhancing Individual Practice

There is a wide range of actual educational value in "educational games". There is a certain amount of trade-off[2] between "education" and "game". When designing an educational game, this trade-off must be kept in mind. Is the primary goal to educate or entertain?

Some may object to the existence of a trade-off, arguing that "learning is fun!". There are two counter-arguments to this: first, the main problem with learning a musical instrument is training muscle control. No matter how exciting and amusing a music teacher makes the new lessons, at some point the students must practice their physical exercises.[3] Second, if learning a musical instrument is so fun, why do music teachers need to force students to practice their scales and exercises? Many children choose to spend time playing computer games or watching television instead of practicing their instrument.

When designing educational games whose primary purpose is to enhance individual practice, the following goals should be considered:

Focus: When the student is playing her instrument, she should be concentrating on her instrument and the sound, not on the technology.

Reliability: A program which assigns grades to students should be as reliable as possible. If a problem is detected while calculating the grade, the user should be notified that the grade may not accurately reflect her ability.

Transparency: Users should be able to determine why the program produced the output it did.

Simplicity: The exercises and algorithms used should be as simple as possible.

These goals are discussed in detail below.

Focus

No interactive tool should distract the student from listening to herself. Developing realtime self-analysis skills – maintaining a constant feedback cycle between sound

[2]This mirrors the trade-off between security and usability. The two are not exact opposites, but there is still some conflict between them. For example, locking and unlocking a car increase the car's security, but slightly reduces the ease-of-use by requiring the user to keep the car's key. In this case, the security benefit far outweighs the usability penalty.

[3]Some people may still argue that "physical training is fun!", but the widespread use of mobile music devices and televisions in fitness clubs suggests that many people do *not* find physical training entertaining by itself.

and action – is absolutely vital for playing a musical instrument. Most instruments have variable pitch (even an instrument with relatively-fixed pitch, such as a saxophone, can play bad pitches due to air flow) and immensely variable sound quality. Music students must learn to adjust their bodies (be it fingers, arm positions, air flow, lip position, etc) automatically in response to the sounds they produce.

Consider an analogy to a baby learning to walk. Standing on two legs is a remarkable feat of balance; babies must learn to make thousands of tiny adjustments to their bodies in response to sensations in their inner ear. Now suppose that we displayed some colors on a computer screen for the baby – blue if the baby was leaning too far forwards, red if they were leaning backwards, etc. A baby might learn to associate the visual colors with her ability to remain upright, instead of using the sensations in her inner ear. Once we remove the computer display, the baby would lose the ability to walk.

For this reason, we suggest that realtime interactive tools should be used with caution. In general, computer analysis and interaction should occur *after* a student has finished playing her instrument. The computer should be used to confirm (or correct) a student's judgment, not as a replacement for realtime self-analysis. This may come as a disappointment to researchers interested in pushing the boundaries of digital signal analysis – realtime processing is much more challenging than offline processing, after all – but realtime multimedia tools may be counter-productive if used indiscriminately.

It should be clear that we suggest caution, not a complete ban. There are some cases where it may be appropriate to introduce technological aids for short-term gain at the expense of long-term development. For example, many violin teachers place pieces of tape on the instruments of young students to show them where to place their fingers – in effect, adding frets to a fret-less instrument. Although violinists must eventually learn how to play without tape, many teachers feel that using tape for a few months is a worthy trade-off. This belief is not universally shared; some prominent violinists argue against using tape. In the same way, a particular realtime computer program may be useful in the short term despite distracting the student from concentrating on her sound.

There is one area which is safe for almost indiscriminate use of realtime tools, however: technical exercises. As discussed in Section 1.1.2, these are short exercises which are aimed at developing specific skills. A realtime visualization tool which is used for specific technical exercises is unlikely to subvert a student's long-term development of her realtime self-analysis skills – especially since different technical

exercises will probably use different visualization techniques. It is unlikely that a student will use one particular technical exercise tool often enough that this tool replaces her own ears.

Finally, it should be emphasized that our caution refers only to realtime interactive tools – programs which provide feedback to the student while the student is playing their instrument. We have no concerns about multimedia tools which provide feedback after the student has finished playing her instrument (i.e. in the "repeat exercise or begin new exercise" part of Figure 1.2), even if this data is gathered via realtime digital signal processing algorithms.

Reliability

Music transcription is a serious problem for CAMIT projects. Before a computer can give feedback about a student's performance, it must recognize music events in the audio stream. However, we lack a general music transcription algorithm. We can avoid this problem by using electronic instruments such as a MIDI piano keyboard. Most previous work on CAMIT therefore focuses on piano – although even then, our transcription is not perfect. Tracking multiple voices in highly polyphonic scores remains problematic.

Violins and other orchestral stringed instruments have their own difficulties for computer music transcription. Double stops, in which two notes are played simultaneously, are common; some violin sonatas include as many as six-note chords.[4] Pitch-detection algorithms for multiple pitches are improving, but there is still work to be done.

However, the biggest difficulty is note segmentation: where does one note begin and the previous note end? Violins playing *pizzicato* have very sharp attacks and exponential decays; violins playing *piano espressivo* have very gradual attacks and almost no decay at all. Even humans have difficulty identifying the exact moment of attack; an individual's judgment of the perceptual attack time can vary enormously [46], so it is no wonder that this remains an open problem.

If a problem occurs with the transcription, then a student's grade will not accurately reflect her actual ability. If possible, the transcription should include some tests – for example, if the transcribed music does not contain the same number of notes that the student was asked to play, a warning should be generated. A student

[4]These are found in advanced works, such as Ysaÿe's 2^{nd} violin sonata. Strictly speaking, a violinist cannot play more than two notes at once. However, moderately skilled violinists can create the illusion of playing three or four notes at once by "rolling" the chord; highly skilled violinists can create the illusion of playing five or six notes at once using a similar technique.

may play the wrong notes, and will almost certainly play notes out of tune, but playing the wrong number of notes is quite rare.

Transparency

The CAMIT project should be more than reliable; it should be *seen* to be reliable. If possible, the CAMIT project should be able to "show its work" if asked.

If we present music teachers with "black-box" software which assigns grades to their students, they will (quite reasonably!) be suspicious. Suppose we develop a general violin transcription program, feed it audio from two students, and then announce that their grades were 68.3% and 92.6%. The first student may reasonably ask for an explanation; if the software cannot justify why the first student received such a poor grade, students and teachers will lose faith in the program.

Many music teachers are suspicious of software that claims to grade music. How can a computer grade beauty, after all? How would a computer recognize the difference between expressive deviations from the expected rhythm and a student with a very loose grasp on rhythm?

To counter this reasonable suspicion, CAMIT projects should be completely open about how grades are determined. Teachers and students should be able to trace the program's behavior, from the basic audio, results of the transcription, intermediate grading steps, to the final grade. After reviewing the program behavior a few times, most users will be content to trust the program and will not bother to check the intermediate steps – but having this ability is still necessary in order to justify that trust.

Simplicity

> "Everything should be made as simple as possible, but no simpler."
>
> - (attributed to) Albert Einstein

The notion of simplicity as a design goal in itself ("Keep It Simple, Stupid") is not new. However, it bears special discussion when applied to CAMIT projects. Simplicity has practical benefits: it can increase the reliability and transparency of programs, making it easier for non-technical music teachers and students to understand how the CAMIT project determines the grades. Simple algorithms also reduce the implementation time, allowing us to create useful CAMIT projects faster.

We should apply the principle of simplicity to our exercise design. As our attempts to create a general music transcription algorithm become more and more complex, it is worth taking a step back and asking ourselves what we really want to do.

It is easy to see how the focus on music transcription came about: we want to create CAMIT projects which help students practice, so we look at how students currently practice, and attempt to create software which can analyze such performances. In fact, less than a year ago I declared that "a CAMIT project that can't grade "Twinkle, twinkle, little star"[5] is useless" to one of my supervisors. However, based on our discussion of technical exercises in Section 1.3, I propose that we re-examine our approach to music transcription for CAMIT projects.

For example, suppose we wish to help students practice their rhythms. We therefore need to gather information about temporal events – specifically, where each note begins and ends. What is the simplest method of detecting onsets? Simply examine the audio samples directly; if a sample is greater than some value, we consider that sample to be an onset. This onset-detection method fails miserably when given audio from violins, but it works quite well for humans clapping. In order to use such a simple transcription algorithm, we would need to ask our students to clap rhythms instead of playing them on their violins.

Is this a reasonable request? As it turns out, classical musicians already *do* practice rhythms by clapping. For some teachers, clapping is the first thing they ask a student to do when that student has difficulty playing a rhythm on her instrument. And invariably when a student cannot perform a rhythm on her instrument, she also cannot clap that rhythm. Removing the instrument allows the student to concentrate fully on the task at hand – learning a difficult rhythm – without extra distractions. Clapping does not tell us where each note ends, of course. However, if the music we are clapping does not contain any rests, then we implicitly know where each clapped "note" "ends" – it must end just before the next clap occurs.

We have now vastly simplified our CAMIT project. Instead of developing, implementing, and testing a complicated note segmentation algorithm for violins, we can rely on a decades-old algorithm (picking peaks directly from the audio samples) to detect sudden loud noises.

The problem of detecting intonation can be simplified. Beginning violin students do not play multiple notes at once, so we can use a simple monophonic pitch detection algorithm. Note segmentation is hard because musicians can play the same note more than once (such as the beginning of 'Twinkle, twinkle'). If a musician never repeated the same note, then note segmentation becomes simple: simply look for areas where the pitch changes.

[5]This is the first piece of music in the Suzuki instrument method books, with which I learned the cello.

Is this a reasonable request? There is very little music which does not repeat notes, so it is not as easily granted as our previous clapping simplification. However, we are not attempting to analyze any arbitrary piece of music, but rather to create a CAMIT project to help students practice their intonation. Do we need to grade *existing* music? No. Can we create short exercises which test intonation, but do not repeat any notes? This is definitely possible. In fact, since repeating a note does not add anything to an intonation test, we would want our intonation exercises to avoid repeated notes even if we *could* reliably perform general violin transcription.

Simple, small-scale targeted technical exercises are also less threatening to a music teacher. We will be the first to admit that there is no chance of computers teaching (and grading!) musical aesthetics in the next fifty years, so music teachers have nothing to fear. However, some music teachers may still be nervous about this prospect when presented with a general CAMIT project. In contrast, no music teacher will feel threatened by a computer program which detects claps and compares them to two bars of rhythmic music.

1.4 Related Work

A number of CAMIT projects have been attempted in the past fifteen years. Work on CAMIT projects may be split into two categories: projects which have a narrowly-focused goal, and projects which attempt to provide a complete learning environment by providing a "virtual teacher" for the students' private practice. A general survey of the use of computers in music education is presented in [5].

1.4.1 Narrowly-focused projects

There have been many CAMIT projects which have a narrow focus. These projects aim to provide one tool which music teachers may use to solve a particular problem.

An excellent example of this focused approach is the work of Robine et al [32]. Saxophone players (of any level of ability) were asked to play five notes – three notes with constant loudness, one note which gradually became louder and then quieter, and one note with vibrato. By analyzing the stability of pitch and amplitude of these five notes, they could accurately predict the overall ability of each student as determined by a professional saxophone teacher. This program may be used by saxophonists to practice their control of airflow. This work is a good example of using multimedia analysis to enhance technical exercises.

Many musicians consider chamber music to be the pinnacle of music: playing music with a few other musicians allows the greatest combination of flexibility and

structure. Oshima et al [29] make chamber music easier for beginning piano students with their Family Ensemble software. This project provides an easy way to play piano duets: non-musicians (such as parents or siblings of a student) may perform the secondary part by arbitrary keys on a piano keyboard. The correct notes are substituted for the secondary (non-musician) player, so this player needs only to play the correct rhythm. In addition, score following techniques are used to compensate for mistakes of the student. Although this work does not provide direct feedback to the student, the extra motivation and sheer 'fun' of music it gives students is extremely valuable.

Research is progressing in developing visualization techniques for music. Ferguson et al [12] investigate ways to present multiple streams of data in a single, easily-understandable display. In [11], Ferguson investigates using realtime sonification to provide feedback to musicians. This may seem counterintuitive – the main focus of a student should be her own sound – but with sufficient care, these techniques may be applied in certain situations.

The PianoFORTE system by Smoliar et al [39] attempts to bridge the gap between simply playing the notes (which MIDI synthesizers can do beyond any human ability) and performing music (by adding the tiny variations in tempo and dynamics which makes music seem "alive"). The system provides visualizations for the dynamics, tempo, articulation, and synchronization (between hands) of a piano performance. This may be used to increase communication between teacher and student by providing easily-viewable representations of these expressive parameters.

A review of four different real-time visual feedback programs for singers is presented in [16]. Some programs displayed only pitch, but others provided feedback about vowel identity and timbre, the latter two aspects being derived from the sound spectrum and amplitude. The most interesting finding is that the amount of information given affects varying skill levels in different ways. This research found that untrained singers benefited the most from a plot of their pitch, while trained singers benefited the most from a keyboard display in which their current pitch is indicated with a highlighted key.

1.4.2 General projects

There are a few large CAMIT projects which aim to provide general instruction: the student plays a complete piece of music, the computer analyzes the performance, and then provides feedback. These systems are primarily intended for self-learning or distance education.

The first major such project was Piano Tutor [8, 9]. This project used score-following software to analyze a student's performance, then used an expert system to judge which mistakes were in greatest need of help. The system then used a combination of graphics, voice, and video to inform the student of these mistakes and how to correct them. The system could also choose simpler tasks for the student, so that students could concentrate on improving specific skills which need help. By using MIDI piano keyboards, researchers avoided the problem of transcribing music and focused on discovering techniques for creating interactive learning tools.

The IMUTUS project [37, 14, 36] is the spiritual successor to Piano Tutor. It is designed to be a complete, autonomous tutor with no human teachers (although the authors note that it will be more successful when used in conjunction with a teacher), but in this case the target instrument is the recorder. This system operates by providing feedback after each performance: the system prioritizes mistakes based on discussions with over 40 recorder teachers. For example, mistakes in articulation are less important for beginning recorder players than control of air flow. To avoid overwhelming the student, the system informs the student of only a few mistakes. Students may also request hints to view extra annotations made by teachers. This project has now ended, but similar work continues with the VEMUS [18, 13] project, now investigating other wind instruments.

i-Maestro [17, 28] also provides an interactive self-learning environment, but this project is also investigating the user of new gesture-based interfaces. An elaborate framework of server software, client software, music exercise authoring tools, P2P techniques, and 3-D motion capture visualization software is planned, to allow students to learn more effectively. This project is still in progress; we look forward to reviewing the project's results.

The Digital Violin Tutor (DVT) [47, 19] provides feedback in the absence of human teachers. DVT offers different visualization modalities – video, "piano roll" graphical displays, 2-D animations of the fingerboard, and even 3-D avatar animations. DVT consists of several interconnected modules. The student's audio is transcribed and compared to the transcription of the teacher's audio. If mistakes are detected, then the proper actions are demonstrated by the 2-D fingerboard animation, video, or the 3-D avatar animation.

1.5 Conclusion

We have discussed the difficulties of learning a musical instrument: practicing the physical motions requires a large number of correct repetitions, but most students

cannot judge whether they have performed those actions correctly. Computers can provide objective analysis in the absence of music teachers.

Computer assistance for practicing technical exercises would be particularly effective. Such exercises may be analyzed without the use of complicated music transcription algorithms, allowing more resources to be spent on the exercise design and visualization, rather than the digital signal processing. We identified four design goals for developing CAMIT software: focus, reliability, transparency, and simplicity.

Chapter 2

Creating Exercises

Generating technical exercises for various levels of playing ability is important for any instrument method book. Technical exercises are short exercises aimed at a particular skill, such as playing long quiet notes or shifting smoothly. These exercises make no attempt at being musical or aesthetically pleasing in themselves; students should focus on the particular skill under development, rather than focusing on aesthetic concerns. To draw an analogy to sports, technical exercises are like weight training. They do not require any strategy, offer little or no entertainment value, do not involve any team-work – but they are the best way of strengthening muscles.

In some cases there may be only one or two exercises, but most of the time students are given a large number of exercises to practice. These exercises differ only slightly from each other, and are organized into levels of increasing difficulty. Writing these exercises by hand can be quite tedious.

This is particularly apparent when we consider CAMIT projects, which could benefit from a library of thousands of exercises. Such a collection would be impractical as physical sheet music, but could be easily duplicated and distributed in digital form. This library could be used to pick material which addresses the specific weaknesses of students – there could be hundreds of exercises simply devoted to strengthening a violinist's fourth finger or switching between registers on a wind instrument. Another use of a large library of exercises is to practice sight-reading. "Sight-reading" is the term for playing a piece of music without previously hearing the work or reading the sheet music in advance. Since each piece of music can be sight-read only once, a large collection of pieces would be needed to practice sight-reading on a regular basis.

To generate a large collection of technical exercises, we therefore turn to Computer-Assisted Composition (CAC) and constraint programming. By expressing the desired characteristics of exercises for each level as a constraint satisfaction problem, a computer can generate thousands of exercises in a matter of seconds.

2.1 Constraint Satisfaction Programming

Modeling compositional tasks with constraint programming is not new. A Constraint Satisfaction Problem (CSP) is defined as a tuple consisting of:

- a finite set of *variables*,
- each associated with a finite *domain*,
- a set of *constraints* which restrict the values that the variables can simultaneously hold.

The task is to assign each variable a value in its domain while fulfilling all constaints.

A good introduction to CSPs is presented in [41]. There have been a few general CSP systems for music composition, including PWConstraints and Situation [2], OMClouds [40], and Strasheela [1]. Specific CSPs have been created; these include rhythmic patterns [34] and instrumental writing [23]. However, these efforts have focused on artistic, rather than pedagogical, goals.

2.2 Creating Rhythmic Exercises

Most musicians have an intuitive sense of what makes a rhythm easier or harder to perform. However, we cannot teach human intuition to a computer – before we can begin creating rhythms with CAC, we must formalize our concept of "easy" and "hard" rhythms. A secondary goal of our formalization of musical knowledge is to keep the mathematics simple enough for interested music teachers to understand – both so that they feel more comfortable using such automatically-created exercises, and so that they may define their own new exercise levels.

2.2.1 Rhythmic Difficulty Levels

Consider music as a series of events. An *event* represents the beginning of a note or rest, regardless of its duration (see Figure 2.1). If this rhythm is clapped (a very common way to practice rhythms), the two bars will sound identical. However, the second bar is harder to read, due to the quarter rest spanning the second beat. In the first bar, there is an event on each beat; this is very useful for students.

Using the terminology of events and durations, different rhythmic difficulty levels can be expressed quite concisely. Table 2.1 shows the levels of rhythmic difficulty used in this thesis. Each level is defined independently, so they may be re-ordered or modified if desired. The rhythmic exercises are quite short: one bar in 4/4 time, which is repeated to create two bars.

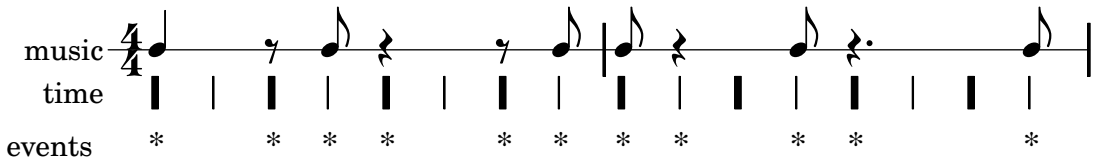


Figure 2.1: Sample rhythm. Time: thick blocks show beats, while thin lines show the eighth note off-beats. Events: a star indicates an event in this position.

The 'tatum' ("Temporal Atom") [4] column represents the number of events into which each quarter-note beat may be divided. For example, to represent sixteenth notes, each beat must be divided into a multiple of 4, and to represent triplet eighths each beat must be divided into a multiple of 3.

Table 2.2 shows the total number of rhythms on each level. In many cases, these rhythms include rhythms which were solutions to some previous levels – for example, level 2 rhythms include all level 1 rhythms, and level 4 rhythms will not necessarily contain any rests. For some applications this may be desirable, but for other applications we may wish to generate only rhythms which specifically include new material. We call these "interesting" or "new" exercises. We therefore introduce extra constraints which exclude uninteresting solutions – for example, level 2 must contain some sixteenth notes, and level 4 must have some rests.

2.2.2 Implementation notes

We used Strasheela [1] and its underlying language Oz [33] to phrase rhythmic exercises as a CSP, and generated sheet music with GNU/LilyPond [27]. Oz is a multi-paradigm programming language, including functional and constraint programming.

Music Representation: Events and Durations

To express a problem as a CSP, we must state what the variables are. Most musical CSPs specify the number of notes at the outset, but this is problematic for more advanced rhythms. One beat may be filled with anywhere from zero (if this beat falls inside a long note) to eight notes (if we allow thirty-second notes).

We therefore create a list of all possible event positions, where each variable represents which event occurs in that position (the "event list"). For example, to produce one bar with notes as fast as eighth notes, we create a list of 8 variables. Each variable represents one possible event – these correspond to the time positions in Figure 2.1.

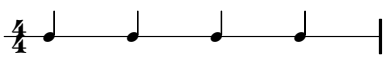
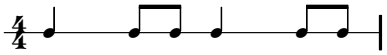
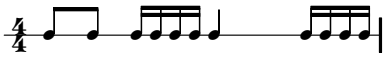
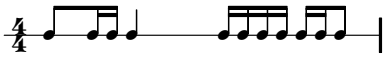
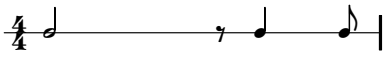
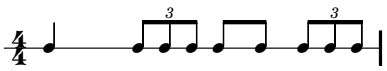
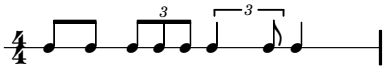
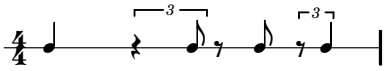
Level	Durs.	Tatum	Events constraints	Example
0	$\frac{1}{4}$	1	Must have an event on every beat, and no rests.	
1	$\frac{1}{4}\,\frac{1}{8}$	2	Must have an event on every beat, and no rests.	
2	$\frac{1}{4}\,\frac{1}{8}\,\frac{1}{16}$	4	Must have an event on every beat, no rests, and each beat is divided into equal durations.	
3	$\frac{1}{4}\,\frac{1}{8}\,\frac{1}{16}$	4	Must have an event on every beat, no rests, and $\frac{1}{16}$ must occur in pairs replacing an $\frac{1}{8}$.	
4	$\frac{1}{4}\,\frac{1}{8}$	2	Must have an event on every beat, and rests can occur only on beats.	
5	$\frac{1}{2}\,\frac{3}{8}\,\frac{1}{4}\,\frac{1}{8}$	2	Must have an event on every second beat, and rests can occur only on beats.	
6	$\frac{1}{4}\,\frac{1}{8}\,\frac{1}{12}$	6	Must have an event on every beat, no rests, and each beat is divided into equal durations.	
7	$\frac{1}{4}\,\frac{1}{6}\,\frac{1}{8}\,\frac{1}{12}$	6	Must have an event on every beat, and no rests.	
8	$\frac{1}{4}\,\frac{1}{6}\,\frac{1}{8}$ $\frac{1}{12}\,\frac{1}{16}$	12	Must have an event on every beat, no rests, and $\frac{1}{16}$ must occur in pairs replacing an $\frac{1}{8}$.	
9	$\frac{1}{2}\,\frac{3}{8}\,\frac{1}{4}\,\frac{1}{8}$	6	Must have an event on every second beat, and rests can occur only on beats.	
10	$\frac{1}{2}\,\frac{7}{16}\,\frac{3}{8}\,\frac{1}{4}$ $\frac{3}{16}\,\frac{1}{8}\,\frac{1}{16}$	4	Must have an event on every second beat, rests can occur only on beats, and a note longer than one beat must either begin or end a beat.	

Table 2.1: Levels of rhythmic difficulty: general musical information.

Level	Norm. sols.	Constraints to exclude uninteresting solutions	New sols.
0	1	$\emptyset$	1
1	16	Cannot have three identical durations on adjacent beats.	10
2	81	Cannot have two identical durations on adjacent beats, and must have at least two $\frac{1}{16}$ on beats.	10
3	625	Cannot have two identical durations on adjacent beats, must have at least six $\frac{1}{16}$, at least two $\frac{1}{8}$, at least one $\frac{1}{4}$, cannot contain any beat with two $\frac{1}{8}$, and must contain at least one beat with a mixture of $\frac{1}{8}$ and $\frac{1}{16}$.	30
4	256	Cannot have three identical durations on adjacent beats, cannot have adjacent rests, and must contain rests of $\frac{1}{4}$ and $\frac{1}{8}$ durations, and must contain notes of $\frac{1}{4}$ and $\frac{1}{8}$ durations.	29
5	576	Cannot have two identical durations on adjacent beats, cannot have adjacent rests, must have at least one note longer than $\frac{1}{4}$ or a $\frac{1}{4}$ tied over a beat, at least one rest, at least two notes, and at most one note longer than $\frac{1}{4}$.	44
6	81	Must have an adjacent $\frac{1}{6}$ and $\frac{1}{8}$, at least one $\frac{1}{4}$, and at least two beats starting with a $\frac{1}{6}$.	10
7	625	Cannot have two identical durations on adjacent beats, must have an adjacent $\frac{1}{8}$ and $\frac{1}{3}$ or $\frac{1}{6}$, at least one $\frac{1}{3}$, at least three $\frac{1}{6}$, at least one $\frac{1}{8}$, and at least one $\frac{1}{4}$.	32
8	4096	Cannot have two identical durations on adjacent beats, must have have an adjacent $\frac{1}{8}$ and $\frac{1}{3}$ or $\frac{1}{6}$, at least one $\frac{1}{3}$, at least one $\frac{1}{8}$, at least two $\frac{1}{8}$, at least one $\frac{1}{16}$, and a $\frac{1}{4}$ on the second or third beat.	64
9	10,000	Cannot have two identical durations on adjacent beats, cannot have adjacent rests, must have an adjacent $\frac{1}{8}$ and $\frac{1}{3}$ or $\frac{1}{6}$, at least one $\frac{1}{8}$, at least one rest with duration $\frac{1}{3}$ or $\frac{1}{6}$, at least one rest with duration $\frac{1}{8}$, at least four notes, and a $\frac{1}{4}$ which is not on the last beat.	28
10	128,164	Cannot have two identical durations on adjacent beats, cannot have adjacent rests, has no event on the beginning of the second and fourth beats, has at least one rest, at least three notes, has exactly one $\frac{1}{16}$, at most one $\frac{1}{4}$, at most two $\frac{3}{4}$, and at most one $\frac{1}{8}$.	84

Table 2.2: Levels of rhythmic difficulty: creating "interesting" exercises. The number of solutions was determined by generating all possible solutions in Oz.

For ease of programming, we represented each event with an arbitrary integer. Figure 2.2 shows our event list.

0: This represents no event occurring at this position.

1: This represents the beginning of a note.

2: This represents the beginning of a rest.

The event list can express any rhythm which fits into our "grid" of beats and beat divisions. However, there are some constraints which we cannot easily express, such as "must contain at least one half note". To constrain the durations directly, we introduce a second list of variables to store durations. The duration list D has the same number of variables as the event list E, and is linked to the event list so that information may pass between the two. The lists are indexed starting from 1.

- $D_i = 0 \iff E_i = 0$
- $D_i = 1 \iff (E_i \neq 0) \wedge (E_{i+1} \neq 0)$
  e.g., starting from position i, E contains `1 1`.
- $D_i = 2 \iff (E_i \neq 0) \wedge (E_{i+1} = 0) \wedge (E_{i+2} \neq 0)$
  e.g., starting from position i, E contains `2 0 1`.
- $D_i = N, N > 2 \iff (E_i \neq 0) \wedge (E_{i+1} = 0) \wedge$
  $\ldots \wedge (E_{i+n-1} = 0) \wedge (E_{i+n} \neq 0)$
  e.g., starting from position i, E contains `1 0 0 2` (for $M = 3$).

One problem remains: how do we define durations at the end of our lists? For example, if our list contains 8 values, and $D_8 = 1$, then E_8 and E_9 must be $\neq 0$. But E_9 does not exist.

To solve this problem, we add a "courtesy 1" to the end of our Event list, creating the E_9 that we needed. This neatly constrains the possible durations without requiring any special cases in our definition of the Duration list.

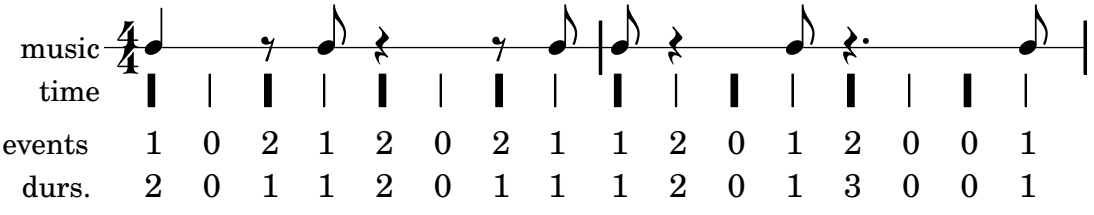


Figure 2.2: Sample rhythm with implementation details.

Defining Constraints

Once we have linked our event and duration lists, implementing the constraints is straightforward – we may express constraints in terms of events and/or durations. We shall examine two levels in detail to illustrate this point.

Level 2 Constraints

To represent quarter notes, eighth notes, and sixteenth notes, we need to divide each beat in 4. Our event and duration lists will therefore contain 16 variables.[1] We begin by specifying that an event occurs on every beat (2.1), and that we do not allow any rests (2.2):

$$\forall E_i | i \in \{1, 5, 9, 13\} : E_i \neq 0 \tag{2.1}$$

$$\forall E_i : E_i \neq 2 \tag{2.2}$$

Now we specify that each beat must be divided into equal durations – in other words, every non-zero duration must be equal to the first duration of that beat (2.3):

$$\begin{aligned} &\forall i | 0 \leq i \leq 7 : \\ &\big((D_{4*i+2} = 0) \quad \vee \quad (D_{4*i+2} = D_{4*i+1})\big) \wedge \\ &\big((D_{4*i+3} = 0) \quad \vee \quad (D_{4*i+3} = D_{4*i+1})\big) \wedge \\ &\big((D_{4*i+4} = 0) \quad \vee \quad (D_{4*i+4} = D_{4*i+1})\big) \end{aligned} \tag{2.3}$$

This constraint may appear to allow invalid solutions for duration sequences such as `1 0 0 1`, but such solutions are already forbidden by the linking between the Event and Duration lists.

Constraints (2.1), (2.2), and (2.3) define level 2. We add more constraints to ensure that we always get "new" exercises: we require at least 8 sixteenth notes (2.4). The boolean values *true* and *false* are represented by 1 and 0, allowing us to add them.

$$(\sum_{D_i}^{32} D_i = 1) \geq 8 \tag{2.4}$$

[1] The Event list will contain 17 elements, but the final element is the "courtesy 1", and thus the value cannot vary.

We cannot have three adjacent identical beats (2.5). Since each beat is subdivided into equal durations, we constrain the only first duration of each beat.

$$\begin{aligned} \forall D_i | i \in \{1, 5, 9, 13\} : \\ (D_i = D_{i+4}) \implies (D_{i+8} \neq D_i) \end{aligned} \tag{2.5}$$

Level 5 Constraints

To create the durations we desire we need to divide each beat into 2, so our lists have 8 variables each.[2] We begin by specifying that we want an event every two beats (2.6):

$$\forall E_i | i \in \{1, 5\} : E_i \neq 0 \tag{2.6}$$

Then we specify that rests can occur only on beats – in other words, if E_i is not a beat, it cannot be a rest (2.7):

$$\forall E_i | i \in \{2, 4, 6, 8\} : E_i \neq 2 \tag{2.7}$$

Constraints (2.6) and (2.7) define level 5. To exclude uninteresting exercises, we add four more constraints: each solution should contain at least one note longer than a quarter note or a tie over a beat (2.8), cannot have identical durations on adjacent beats (2.9), cannot have adjacent rests (2.10), and contains at least two rests and has at most one note longer than $\frac{1}{4}$ (2.11).

$$(D_3 \neq 0) \vee (D_7) \neq 0 \tag{2.8}$$

$$\forall D_i | \in \{1, 3, 5\} : D_i \neq D_{i+2} \tag{2.9}$$

$$\begin{aligned} \forall D_i | \in \{1, 3, 5\} : ((E_i = 2) \wedge (D_i = 2)) \implies (E_{i+2} \neq 2) \\ \forall D_i | \in \{1\} : (E_i = 2) \wedge (D_i = 4)) \implies (E_{i+4} \neq 2) \end{aligned} \tag{2.10}$$

$$(\sum_{E_i}^{16} E_i = 2) \geq 2, \quad (\sum_{E_i}^{16} E_i = 1) \leq 1 \tag{2.11}$$

[2]Again, the Event list actually has 9 elements, but the final element is the fixed "courtesy 1" and is thus not a variable.

2.2.3 Discussion of Rhythmic Exercises

Our music representation offers a good balance between simplicity and capability. Constraints on the events and durations can be understood – and more importantly, *created* – by music teachers with even fairly little programming ability. If a music teacher in a different tradition (e.g., Cuban drumming or Irish folk tunes) wished to create rhythmic exercises tailored for her particular needs, she should be able to do so with relatively little confusion.

As previously mentioned, a CSP must specify the number of variables before searching for a solution. For our goal of creating short rhythmic exercises, it was sufficient to create a list of Events and Durations. This approach fixes the total duration (in our case, to one bar), but allows the number of notes to fluctuate. Another approach would be to fix the number of notes, and let the total duration vary – exercises with many sixteenth notes may only be one bar long, while an exercise with half notes may be six bars long. A more sophisticated solution would be to fix the total number of notes, but allow notes with a duration of 0. Such notes would be considered "non-existing", and would not be exported into musical notation. This method would allow both the total duration, and total number of "existing notes", to vary between different solutions. This is covered in greater detail in [1].

The simple rhythms produced in our exercises do not use any tied notes. However, our music representation may be easily extended to create rhythms which include ties: let 3 in the event list represent "begin a note which is tied to the previous note". We also need the constraint that 3 cannot follow 2 (you cannot tie a rest to a note). With those small modifications, we can begin writing constraints to allow ties in any position we desire.

Adding levels which contain different time signatures is another natural step. This poses no problem for our music representation; we simply set the length of the Event and Duration lists according to the total number of beats and the tatum. However, our music representation cannot accommodate arbitrarily changing time signatures. If we know in advance that a certain level will contain one bar each of $\frac{2}{4}$, $\frac{3}{4}$, and $\frac{4}{4}$ time, then there is no problem (since the total number of beats is still fixed), but we cannot create levels which include an unknown number of bars of unknown time signatures without using the 0-duration trick mentioned above.

Some of our "interesting" constraints may seem a bit arbitrary – for example, levels 8 and 9 constrain on which beat a quarter note can appear. There is no musical reason for constraints this tight; rather, this was motivated by the intended use of these exercises. These exercises are to be downloaded along with a CAMIT program.

In order to reduce the download size of the data files, the number of exercises was tightly constrained. This is discussed in more detail in Section 4.2.2.

The order of exercise difficulty levels is based on our intuitions about rhythmic difficulty. We are not aware of any definitive work in music education which lays out the best order to introduce rhythms (e.g. Should sixteenth notes be covered before rests? Are dotted quarter notes easier or harder than triplets?). There was little uniformity amongst different instrument method books we examined, so we drew upon our own intuitions. This is one area of future research: we should produce a few different orderings of these rhythmic difficulty levels, and track the progress of new musicians through the levels. Do most students get stuck on the same level(s)? Is there a difference in student progress after six weeks?

2.3 Creating Violin Intonation Exercises

The difficulty of performing simple rhythms is relatively constant across instruments. Fast notes are more challenging on certain instruments (for example, a beginner can play sixteenth notes on a violin much easier than on a double bass), but in general, rhythmic difficulty increases equally over all instruments.

The difficulty of performing *pitched* music varies tremendously. For example, playing a D major scale is simple on both violin and voice – but if we change the scale to C♯ major, it becomes considerably harder on the cello, while remaining at the same difficulty for voice. In fact, some singers may not even realize that the scale started on a C♯ instead of a D! Conversely, some exercises are very easy on a violin, while being difficult or impossible for singers. Figure 2.3 shows one such example.

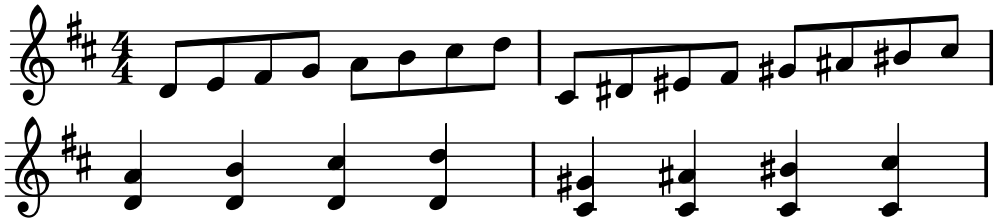


Figure 2.3: Violin vs. singer exercise difficulty: the first two bars are equally easy for singers, but a beginning violinist will find the second bar considerably harder than the first bar. The third and fourth bars are impossible for singers, while the third bar is trivial for beginning violinists and the fourth bar is quite challenging.

2.3.1 Violin Intonation Difficulty Levels

Our formulation of the difficulty of intonation exercises will therefore be very specific to a particular instrument – in this case, the violin. There is enough similarity with

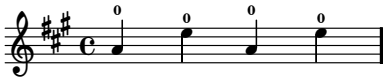
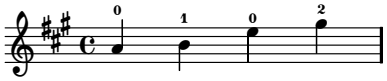
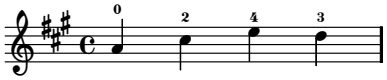
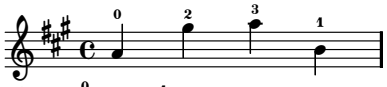
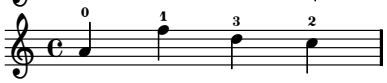
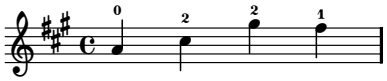
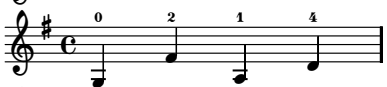
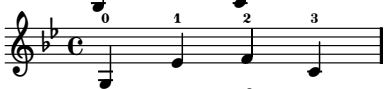
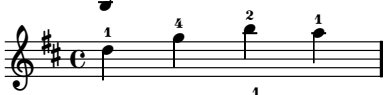
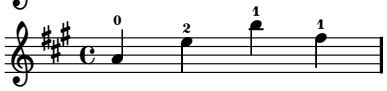
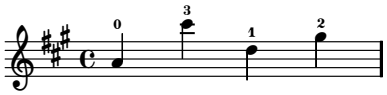
Level	Variable domains[a]	Other constraints	Example
0	S: 1 2, H: 1, F: 0, K: A+	∅	
1	S: 1 2, H: 1, F: 0 1 2, K: A+	Cannot change strings to a fingered note.	
2	S: 1 2, H: 1, F: 0 1 2 3 4, K: A+	Cannot change strings to a fingered note.	
3	S: 1 2, H: 1, F: 0 1 2 3 4, K: A+	No fingered fifths.[b]	
4	S: 1 2, H: 1, F: 0 1 2 3 4, K: A-	No fingered fifths.	
5	S: 1 2, H: 1, F: 0 1 2 3 4, K: A+	∅	
6	S: 3 4, H: 1, F: 0 1 2 3 4, K: G+	∅	
7	S: 3 4, H: 1, F: 0 1 2 3 4, K: G-	∅	
8	S: 1 2, H: 1 3, F: 0 1 2 3 4, K: D+	∅	
9	S: 1 2, H: 1 3, F: 0 1 2 3 4, K: A+	Cannot change fingers or strings while changing positions.	
10	S: 1 2, H: 1 3, F: 0 1 2 3 4, K: A+	∅	

Table 2.3: Levels of violin intonation difficulty: general musical information.

[a] 'S' for string, 'H' for hand positions, 'F' for fingers, and 'K' for the key (+ major or - minor). Setting a key contrains the possible values of the pitches.

[b] "Fingered fifths" is a violin term for playing adjacent notes on adjacent strings with the same finger.

the viola to allow these exercises to be transposed to that instrument.

A violin's pitch is determined by three factors: the *string* that is being played, the *position* that the left hand is in, and the highest *finger* that is touching the string. In a strictly acoustic sense, these last two items are merged into one factor: the length of the vibrating string. However, violinists are taught to think in terms of placing fingers within a certain hand position. When creating exercises, we must certainly consider which fingers are being used, and whether the student must move her left hand ("shifting positions") in order to reach a certain note.

Table 2.3 shows our levels of intonation exercises for violin. They correspond approximately to the pitches covered in the first year of many violin methods. Each

Level	Normal sols.	Constraints to exclude uninteresting solutions	New sols.
0	2	First note is open A.	1
1	86	First note is open A, at least one finger 1, and at least one finger 2.	10
2	686	First note is open A, at least one finger 3, and at least one finger 4.	20
3	4204	First note is open A, two notes each on E and A strings, and at most one finger 0.	108
4	4204	First note is open A, two notes each on E and A strings, at most one finger 0, at most two fingers each of 1-4, at least one note found in a minor scale not found in a major scale.	94
5	6056	First note is open A, at most one finger 0, at most two fingers each of 1-4, and at least one fingered fifth.	96
6	6056	First note is open G, at most one finger 0, at most two fingers each of 1-4, and two notes each on D and G strings.	156
7	6056	First note is open G, at most one finger 0, at most two fingers each of 1-4, two notes each on D and G strings, and at least one note found in a minor scale not also found in a major scale.	134
8	3710	First note is the D on the A string, no open strings, at most two fingers each of 1-4, and at least two string changes.	102
9	39,948	First note is open A, at most one finger 0, at least two notes each in first and third positions, and cannot shift down to 3^{rd} or 4^{th} finger.	93
10	64,982	First note is open A, at most one finger 0, at two notes each in first and third positions, cannot shift down to 3^{rd} or 4^{th} finger, and cannot change positions without changing fingers or strings.	137

Table 2.4: Levels of violin intonation difficulty: creating "interesting" exercises.

exercise contains four quarter notes – we want beginners to be focused on their finger placement, not distracted by any rhythmic problems. In addition to constraining the violinist's strings, positions, and fingers, each exercise level is in a specific key, constraining the pitches.

As with the rhythmic difficulty levels, we shall repeat each exercise to create two-bar exercises; this gives students two chances to attempt the exercise. In addition, extra constraints are added to create "interesting" exercises. These constraints are presented in Table 2.4. In order to improve the violinist's intonation, the first note is always the tonic of the key.

2.3.2 Implementation notes

Since the difficulty of intonation exercises depends on how a violinist would actually play her instrument, their creation is more challenging than rhythmic exercises.

Music representation: Notes, Strings, Hand positions, and Fingers

Our intonation exercises contain four notes, with each note being defined by its Pitch, String, Hand position, and Finger. We therefore create four lists, with four elements in each list.

Pitch (P_i) is represented with MIDI pitch values, which may theoretically be any integer from 0 to 127. A beginning violinist will not exceed the range 55 (open G string) to 88 (one octave above the open E string), but we do not need to constrain these values manually – this is handled by the linking of our lists.

String (S_i) can be four values: 1 (the E string) to 4 (the G string).

Hand position (H_i) refers to the left hand position. Unfortunately, the violin terminology for positions does not map easily onto integers. The normal hand position is called "first position", but a lower "half position" exists. In addition, many position names allows the first finger to be placed in two different places. For example, "third position" places the first finger on either a D or $D\sharp$ on the A string, while "fifth position" places the first finger on either a F or $F\sharp$.

We therefore use this variable to store the position of the first finger, counting in semitones above the open string. "Half position" is then position 1, "First position" is position 2, and "third position" is either position 5 or 6.

Finger (F_i) is the finger number. Violinists[3] start counting with the index finger as 1. Just as positions do not uniquely identify a note in violin terminology, the exact pitch that each finger plays may vary. For example, in first position on the A string, the 2^{nd} finger may play both C and $C\sharp$. Our music representation handles this ambiguity in the way we link our lists together.

The String, Hand position, and Finger lists do not influence each other – a violinist may use any finger, in any position, on any string. Violinists rarely play in high positions on the G string, but it is not unheard of. Individual exercise levels may

[3]Pianists count with the thumb being "finger 1", which causes considerable confusion for students studying piano and violin at the same time.

constrain these values, but they should not be constrained by the general music representation.

Values in the Pitch list *are* influenced by the other lists, as shown in (2.12). This makes use of G_i, which is the exact number of semitones a finger is placed above the Hand position. The definition of G_i is shown in (2.13), and requires Oz's ability to work with partially-determined variables.

$$P_i = \begin{cases} 83 - (7 * S_i) & \text{if } F_i = 0 \\ 83 - (7 * S_i) + (H_i - 2) + G_i & \text{if } F_i > 0 \end{cases} \tag{2.12}$$

$$G_i = \begin{cases} 0 & \text{if } F_i = 0 \\ (1 \vee 2) & \text{if } F_i = 1 \\ (3 \vee 5) & \text{if } F_i = 2 \\ (5 \vee 6) & \text{if } F_i = 3 \\ 7 & \text{if } F_i = 4 \end{cases} \tag{2.13}$$

This implicitly sets the range of possible pitches. Since S_i can be at most 4, this sets the lower limit of pitches to be 55. If we limit the range of P_i (which we do in every exercise level), the upper limit is set accordingly.

We could change this music representation to cover the viola instead of violin by simply changing the constant from 83 to 76. Representing the cello requires a constant of 64, as well as modifying the definition of G_i – fingers on a cello cannot cover the same range as fingers on a violin.

2.3.3 Definition Constraints

One constraint is applied to all levels: in order to simplify our audio analysis, we require that adjacent notes have different pitches (2.14).

$$\forall i \in \{1, 2, 3\} : P_i \neq P_{i+1} \tag{2.14}$$

We shall examine two levels for examples of the other constraints.

Level 4

This level specifies the domains of our variables (2.15). In addition, we specify a particular key (2.16) by disallowing scale degrees which do not appear in minor. To specify the type of minor, we let $T = 69$ (MIDI pitch for 'A').

$$\begin{aligned} \forall i \in \{1,2,3,4\}: \quad & S_i \in \{1,2\} \\ \forall i \in \{1,2,3,4\}: \quad & H_i = 2 \\ \forall i \in \{1,2,3,4\}: \quad & F_i \in \{0,1,2,3,4\} \end{aligned} \tag{2.15}$$

$$\forall i \in \{1,2,3,4\} : ((P_i - T) \mod 12) \notin \{1,4,6,9,11\} \tag{2.16}$$

We also disallow fingered fifths (2.17).

$$\forall i \in \{1,2,3\} : (S_i \neq S_{i+1}) \implies \big((F_i = 0) \vee (F_{i+1} = 0) \vee (F_i \neq F_{i+1})\big) \tag{2.17}$$

These constraints determine difficulty level 4, but we add extra constraints to create "interesting" exercises. Specifying the number of notes on each string (2.18) and the maximum number of times that each finger can be used (2.19) is straightforward. We also require that each exercise contain at least one note found in a minor scale which is not found in a major scale (2.20).

$$\big(\sum_{i=i}^{4} S_i = 1\big) = 2, \quad \big(\sum_{i=i}^{4} S_i = 2\big) = 2 \tag{2.18}$$

$$\begin{gathered} \big(\sum_{i=i}^{4} F_i = 0\big) \leq 1, \\ \big(\sum_{i=i}^{4} F_i = 1\big) \leq 2, \quad \big(\sum_{i=i}^{4} F_i = 2\big) \leq 2, \\ \big(\sum_{i=i}^{4} F_i = 3\big) \leq 2, \quad \big(\sum_{i=i}^{4} F_i = 4\big) \leq 2 \end{gathered} \tag{2.19}$$

$$\begin{pmatrix} \big(\sum_{i=1}^{4}((P_i - T) \mod 12) = 3\big) \vee \\ \big(\sum_{i=1}^{4}((P_i - T) \mod 12) = 8\big) \vee \\ \big(\sum_{i=1}^{4}((P_i - T) \mod 12) = 10\big) \end{pmatrix} > 0 \tag{2.20}$$

Level 9

In addition to the variable domains (2.21) and key (2.22) ($T = 69$), this level is defined by same-finger shifting (2.23).

$$\begin{aligned} \forall i \in \{1,2,3,4\}: &\quad S_i \in \{1,2\} \\ \forall i \in \{1,2,3,4\}: &\quad H_i \in \{2,5\} \\ \forall i \in \{1,2,3,4\}: &\quad F_i \in \{0,1,2,3,4\} \end{aligned} \tag{2.21}$$

$$\forall i \in \{1,2,3,4\} : ((P_i - T) \mod 12) \notin \{1,3,6,8,1\} \tag{2.22}$$

$$\begin{gathered} \forall i \in \{1,2,3\} : (P_i \neq P_{i+1}) \implies \\ \big(\big((F_i = 0) \vee (F_{i+1} = 0)\big) \vee \big((F_i = F_{i+1}) \wedge (S_i = S_{i+1})\big)\big) \end{gathered} \tag{2.23}$$

To create "interesting" exercises, we add the simple constraints that there be two notes each in first and third positions (2.24), at most one finger 0 (2.25). We also specify that we cannot shift down to 3^{rd} or 4^{th} finger (2.26).

$$\big(\sum_{i=i}^{4} P_i = 2\big) = 2, \quad \big(\sum_{i=i}^{4} P_i = 5\big) = 2 \tag{2.24}$$

$$\big(\sum_{i=i}^{4} F_i = 0\big) \leq 1 \tag{2.25}$$

$$\begin{aligned} \forall i \in \{2,3,4\}: &\quad (P_i < P_{i-1}) \implies (F_i \neq 3) \\ \forall i \in \{2,3,4\}: &\quad (P_i < P_{i-1}) \implies (F_i \neq 4) \end{aligned} \tag{2.26}$$

2.3.4 Discussion of Intonation Exercises

Our exercises are extremely limited. The first note is almost entirely fixed[4] in order to give violinists a "reference pitch". In addition, this first pitch is taken as the tonic of the scale if users choose to use just intonation instead of equal temperament. This is discussed more in Section 3.4.1 and Section 4.3.2. We are therefore choosing only

[4]The pitch, string, and finger is always fixed, but it may be played in any position.

three notes. We have further limited the number of possible exercises by restricting each exercise to only two strings. Despite these enormous restrictions, we have still reached sixty-five thousand exercises for level ten.

To emphasize the number of potential exercises, consider a level in which we create 8 notes, with each note being on any string, first position, allowing all four fingers but no open strings, in G major. This gives $16 \cdot 15^7 \approx 3$ billion different exercises; clearly beyond the practical limits of computer-generated music. We can reduce this number tremendously by adding "interesting" constraints, but at a some point these constraints become ridiculously specific.

Future work in this area should not generate all possible exercises. Instead, it should randomly generate a set of exercises for the near future (where the set would probably range from 10 to 100 exercises, depending on its intended use). Once this random exercise generation is implemented, we could eliminate the specific exercise levels. Instead, we could maintain a set of constraints, which would be modified as the computer or instructor sees fit. For example, instead of level 2 introducing all four fingers and level 6 introducing the D and G strings, we could begin from $F \in \{0\}$, $S \in \{1, 2\}$ and gradually increase the domains. The program would track the user's intonation on each finger and each string, and increase the domain of each list when the student's estimated skill was sufficient.

This would involve sophisticated audio analysis and possibly machine learning – instead of calculating a single number to represent the intonation error for each note, the computer would need to estimate the amount of intonation error due to string, position, and finger. For example, consider a student whose 1^{st} and 2^{nd} fingers were in tune on the E, A, and D strings, but whose 3^{rd} finger was flat on all strings. In this case, the software would begin giving the student exercises on the G string (since the strings were not a problem), but would not introduce the 4^{th} finger.

In addition to removing or modifying constraints to control what technical skills were being tested, such a system could create exercises truly targeted to the current user. In the above example, the software could add a constraint that at least two notes should involve the 3^{rd} finger. This constraint would remain until the user's 3^{rd} finger improved, at which point the software may return to having an equal chance of any finger, or may introduce the 4^{th} finger.

The order of these intonation difficulty levels is based on our own intuitions. Just as we found that the order of rhythmic material varied amongst different method books, we discovered that the order of new pitches varied. Some books introduced shifting before covering minor scales and accidentals; other books began with the

student playing on the D and A strings instead of the A and E strings, etc. In the absence of a definitive answer from music education texts, we used our own judgement to put intonation exercises into an order. Future work could include experiments with beginning violin students, testing a few different orders of material to determine which order produced the best results.

2.4 Conclusion

Before asking students to perform exercises, we must have exercises to give them. Writing exercises by hand is certainly possible, but this introduces practical limits to the number of exercises. In this chapter, we introduced methods of creating exercises automatically with Constraint Satisfaction Programming.

More importantly, the definition of these exercise levels is simple enough that interested music teachers may create new levels on their own. Given the wide range of technical exercises – different ability levels, different instruments, different musical styles – allowing users with relatively little programmering ability to define their own levels is crucial for widespread use.

Chapter 3

Exercise Analysis

All CAMIT projects must analyze the performances of students. If the student played the exercise on a MIDI instrument (such as many piano CAMIT projects), no audio analysis is necessary. In our case, we must analyze the audio to extract temporal or pitch information before grading the performance.

Our audio analysis ("music transcription") is relatively straightforward. For the rhythmic exercises, we must detect a series of claps. For the violin intonation exercises, we must detect the violin's pitch and the moments at which it changes notes. As per our design goals described in Section 1.3.3, we deliberately used the simplest transcription algorithms which yielded accurate results.

Once we have extracted the musical information from our raw audio, we must grade each performance. The primary purpose of grading is to determine if the student is ready to attempt harder exercises, while the secondary purpose is to help the student identify and fix problems. The latter goal is discussed in Section 4.2.1 and Section 4.3.1.

3.1 Music Transcription: Rhythmic Exercises

The automatic detection of onsets in music has been investigated in great detail, ranging from transcribing percussive music [35] to polyphonic music with varying instruments [3, 21]. Our task, however, is much simpler: detecting claps in monophonic audio, where each clap is at least 100 ms after the previous clap.

To detect claps, we split the audio stream into frames of 512 samples and take the RMS amplitude of each frame. We then normalize the frames so that the largest value is 1.0, and look for peaks in the result with values greater than a threshold K_t (generally $K_t = 0.1$ or 0.2), and which occur at least 9 frames after a previously-detected peak.

3.1.1 Onset Detection

Digital audio on computers is generally represented by 16-bit samples at 44,100 Hz. This produces a huge amount of numbers to analyze – a typical clapping exercise such as Figure 3.1 will usually be at least 10 seconds long, generating 441,000 numbers between 0 and 65,535.

Figure 3.1: Sample clapping exercise.

We begin by reducing this number: we group these samples into frames of 512 samples. Each frame is approximately 11.6 milliseconds long. We calculate the value of each frame by taking the Root-Mean-Square of the samples (3.1).

$$F_{rms} = \sqrt{\frac{{s_1}^2 + {s_2}^2 + ... + {s_n}^2}{n}} \tag{3.1}$$

where F_{rms} is the value of the frame, and s_1 through s_n are the values of the individual samples assigned to that frame. In this case, $n = 512$.

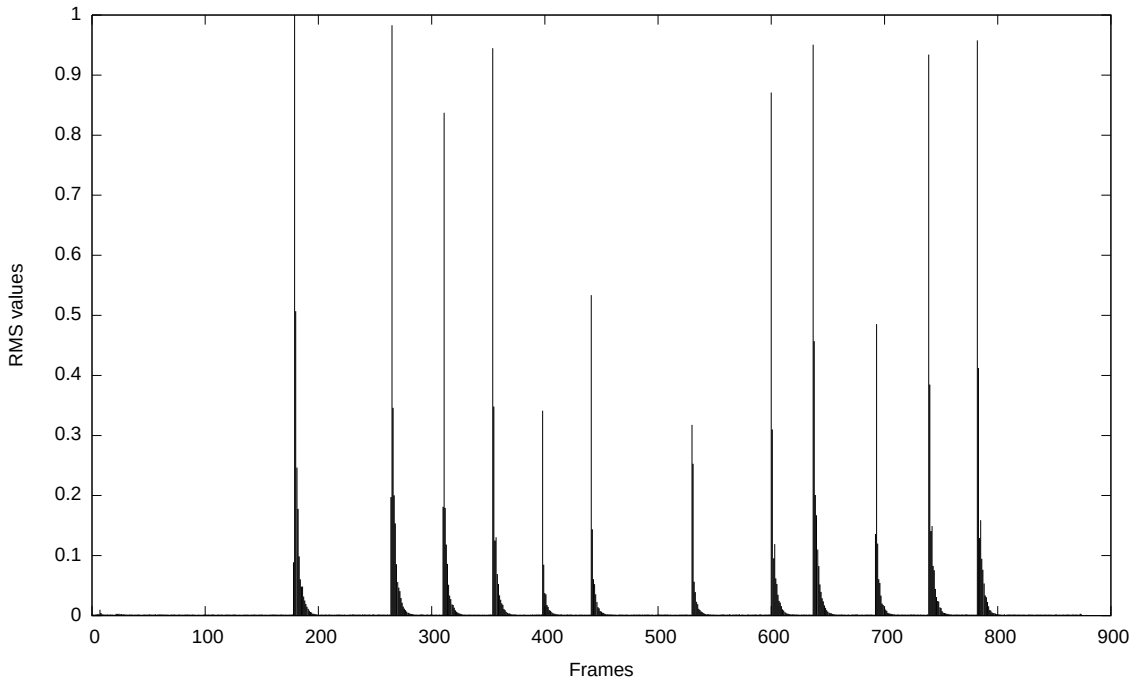
Typical results are shown in Figure 3.2a. The claps are clearly visible to human eyes. To detect them automatically, we look for peaks whose values are greater than a threshold K_t. Occasionally some artifacts of recording or an echo will produce an additional peak. In Figure 3.2a, there is such a peak 4 frames after the final large peak. To avoid such problems, we discard any peak which is within 9 frames (104 milliseconds) of a previous peak.

3.1.2 Discussion of Rhythm Transcription

Our onset detection is extremely simple, but quite effective for claps. In a quiet room, there is no difficulty in detecting normal claps. It is possible to fool the detector by clapping once or twice as loudly as possible and then clapping as quietly as possible, but this requires special effort.

This onset detection will not function in a room with significant background noise, such as an elementary school classroom or a party. A human clapping once or twice will have no difficulty being detected, but when clapping sixteenth notes, the amplitude of the claps varies enormously, and cannot be reliably detected.

We considered adding a tapping mode: instead of clapping, a user would tap on a mouse button or the space bar. The tapping would be entirely unaffected by the background noise, making it suitable for a party game. However, this could introduce



(a) RMS values

Onsets: `179 265 311 354 398 441 530 600 637 693 739 782`

(b) Onsets detected

Figure 3.2: Transcription of a typical clapping exercise.

other problems: although many users may find timing key presses or mouse clicks quite easy,[1] some users may have problems with this. Clapping is very simple – nobody has any doubt about when a clap occurred, and most humans can clap.

3.2 Grading Rhythms

When creating an algorithm for automatically grading rhythmic exercises, we consider the following goals:

1. The closer a student gets to the correct answer, the higher the grade should be. However, we should not expect students to clap in precisely the correct frame.[2]

2. Grades should not vary based on the number of notes in the exercise.

3. A student should fail the exercise if she claps the wrong rhythm (i.e. adds an extra clap, misses a clap, or claps in significantly different places).

[1] Especially users under the age of 30, who began playing computer games before they were ten years old.

[2] Each frame is less than 12 milliseconds long. Such accuracy would be pushing the bounds of human ability (although not impossible), and is certainly not required for music.

These goals are satisfied with the the well-known Mean-Squared-Error (MSE) function (3.2).

$$\text{Error} = \frac{1}{n} \cdot \sum_{i=1}^{n} (D_i - E_i)^2 \tag{3.2}$$

where D_i and E_i are the detected and expected onsets, and n is the number of onsets. Both lists are indexed starting from 1.

3.2.1 Aligning Onsets

Students do not begin clapping at precisely the moment that we begin recording, so the detected onsets do not begin at 0. Typical values are shown in Figure 3.3.

Detected:	179	265	311	354	398	441	530	600	637	693	739	782
Expected:	0	86	129	172	215	258	344	430	473	516	559	602

Figure 3.3: Comparison of typical detected onsets and expected onsets.

To align the detected onsets with the expected onsets, we must find an offset x such that $\sum (D_i + x - E(i))^2$ is minimized. Minimizing the MSE is a well-known problem; it occurs when the derivative is 0. The solution is derived in (3.3).

$$\begin{aligned}
0 = &\quad \tfrac{d}{dx} \textstyle\sum_{i=1}^{n} (D_i + x - E_i)^2 \\
= &\quad \textstyle\sum_{i=1}^{n} \tfrac{d}{dx} (D_i + x - E_i)^2 \\
= &\quad \textstyle\sum_{i=1}^{n} 2 \cdot (D_i + x - E_i) \cdot (1) \\
= &\quad \textstyle\sum_{i=1}^{n} D_i + \sum_{i=1}^{n} x - \sum_{i=1}^{n} E_i \\
n \cdot x = &\quad \textstyle\sum_{i=1}^{n} E_i - \sum_{i=1}^{n} D_i \\
x = &\quad \bar{E} - \bar{D}
\end{aligned} \tag{3.3}$$

As long as students clap the correct number of times, we can easily find the optimal offset by comparing the mean values of these lists.

If students do not clap the correct number of times, comparing the mean values will not necessarily yield an optimal alignment. We test two possible alignments: using the offset calculated with (3.3), and setting the offset to the first onset. We take the alignment which produces the lowest error score.

3.2.2 Calculating Rhythm Errors

With (3.3), we can now align our detected onsets. In Figure 3.4, $\bar{E} = 315$, $\bar{D} = 494$, so the ideal offset is -179. We shall use A_i to refer to the aligned detected onsets.

Aligned	0	86	132	175	219	262	351	421	458	514	560	603
Expected	0	86	129	172	215	258	344	430	473	516	559	602
Differences	0	0	3	3	4	4	7	9	15	2	1	1

Figure 3.4: Aligned onsets, expected onsets, and their (absolute) difference.

If we knew that student performances would always be approximately correct, we could apply Equation 3.2 directly. However, suppose that a student misses one note in the middle of the exercise. In this case, for the first half of the exercise, each A_i matches with a E_i, but in the second half, each A_i matches with E_{i+1}. In this case, we wish to give the student a poor mark for missing one note, but we should still recognize that the second half was correct. A similar argument can be made if the student adds an extra clap, where A_i matches with E_{i-1}.

To accommodate such errors, instead of matching E_i with A_i, we match E_i with the closest value in A. In an exercise with a missed clap, this gives a large penalty for one note, instead of a large penalty for many notes. To penalize extra claps, we match A_i with the closest value in E. The complete algorithm is presented in Algorithm 1. After experimentation, the constants were fixed as $K_s = 0.45$ and $K_m = 100$. Applying this algorithm to the exercise in Figure 3.4 produces a final grade of 68.0%.

In some cases, a student may have missed a note yet would still receive a passing grade. For example, missing one sixteenth note in the middle of a series of sixteenths in an exercise with 16 notes in total at 60 BPM would only add 13 to the error sum; if the other notes were perfect, the student would still receive a grade of 87%. To avoid this false passing grade, if length(A) $\neq$ length(E) we reduce the actual calculated grade to a failing grade.

3.2.3 Discussion of Rhythmic Grading

MSE was chosen because it is a standard way to measure errors, and initial tests showed a good relationship between the our intuitive sense[3] of rhythmic accuracy and the grades assigned by Algorithm 1.

[3]We have been performing music for twenty years and have taught music for five years, so our "intuitive sense" is not completely irrelevant. However, it *is* still just intuition.

Algorithm 1 Calculating Rhythmic Errors

Require: A // list of aligned onsets
Require: E // list of expected onsets
Require: K_m // constant: maximum amount of error for each note
Require: K_s // constant: scaling factor for total errors
Require: $f(L, x)$ // finds the closest value to x in list L
  $s \leftarrow 0$ // sum of errors
  for all i in length(E) **do**
    $e \leftarrow |f(A, E_i) - E_i|$
    $e \leftarrow e^2/\text{length}(E)$
    $e \leftarrow \min(e, K_m)$
    $s \leftarrow s + e$
  end for
  for all i in length(A) **do**
    $e \leftarrow |f(E, A_i) - A_i|$
    $e \leftarrow e^2/\text{length}(A)$
    $e \leftarrow \min(e, K_m)$
    $s \leftarrow s + e$
  end for
  return $100 - K_s \cdot s$

It must be noted that (3.3) only produces an optimal alignment if we rely on three assumptions: that the performance contains the correct number of notes, that each A_i matches with each E_i, and that the amount of error per note is not cut off by our maximum value K_m. However, Algorithm 1 was designed to accommodate situations where these assumptions do not hold.

For performances with an incorrect number of notes, our alignment will not necessarily produce the best score. More sophisticated algorithms could certainly be used for aligning exercises with the incorrect number of notes: we could try setting the offset to $D_i - E_j$ for every pair of (i, j), and pick the alignment which produced the best score. Alternately, we could simply try every possible alignment. Most performances will contain approximately one thousand frames, and testing an alignment does not require a great deal of computational power.

However, this conflicts with our overall goal of simplicity as discussed in Section 1.3.3. Students who misread the rhythm to such an extent that they do not clap the correct number of times will not receive a passing grade in any event; it does not matter a great deal whether they receive 25% or 45%. Performing an exercise again takes only 10 seconds; students who misread the rhythm should simply try again.

A similar argument applies to performances in which the nearest value to A_i is not

E_i, and to performances which contain at least one note which triggers the maximum error per note (K_m). (3.3) may return a sub-optimal solution, so a student's grade may be lower than it should be. However, once K_m comes into play, the student will have failed the exercise already.

The constants K_s and K_m were fixed by experimenting with a few test cases and the our intuitions. Future work would include an experiment where music teachers arranged performances of rhythm exercises from best to worst, and then fix the values of K_s and K_m to match their results.

Some rhythm-grading projects use a pass/fail mechanism for each note: if a musician is within x frames, the note is correct; otherwise, the note is incorrect. We decided not to use this method. First, some musicians can reliably clap within 2 frames of the correct time; such accuracy should result in higher scores than a musician who is merely within 5 frames. Second, our grading algorithm distinguishes between "has the right idea" (say, within 10 frames) and "has absolutely no clue". If we used a pass/fail (setting the boundary at 7 frames), then both types of performances would receive a grade of 0%, instead of distinguishing between 30% and 0%.

3.3 Music Transcription: Intonation Detection

The automatic transcription of violin music – even solo violin – is an ongoing area of research. Pitch detection for monophonic violin sounds is reliably performed by modern algorithms such as YIN [10], but polyphonic violin chords require special attention [25]. However, the main problem is detecting the onsets of violin notes: although a violin may occasionally play notes with a sharp attack (e.g., *pizzicato*), most notes have a gradual attack, making automatic transcription difficult. A novel approach to this problem is fusing data from audio and video [42]; surveys of traditional transcription methods are given in [21, 6].

We can avoid both these problems by designing targeted exercises. Our exercises do not include chords, so we do not need to worry about polyphonic pitch detection. We have also specified that our intonation exercises will not contain any repeated notes – in other words, the onset of a new note will always involve a change of pitch. Finally, since our exercises are targeted at beginning violin students, we need not worry about vibrato – violinists only learn vibrato after two or three years of study. Our automatic transcription of violin music need only look at the changes in pitch. This was first attempted by Collins [7], although he was attempting to solve a much harder problem: to segment real music rather than targeted beginner exercises.

3.3.1 F_0 Detection

Strictly speaking, most "pitch detection" algorithms do not actually attempt to detect the pitch. "Pitch" is a psychoacoustic term for a certain human sensation, and cannot be automatically calculated. Some algorithms may perform "pitch estimation," where the algorithm attempts to estimate the pitch that a human would report. However, most algorithms which perform "pitch detection" are in fact "fundamental frequency (f_0) detection" algorithms. The fundamental frequency of a signal is the inverse of the shortest period at which it repeats. Unlike pitch, the fundamental frequency can be determined via physical measurement. In most cases, the fundamental frequency is identical (or at least close) to the pitch.

Real-world audio signals never repeat exactly; even if we examine a small portion of the audio (generally 512 or 1024 samples), an apparently "stable" note will not contain any exact repetitions. For this reason, when detecting the fundamental, we can only pick the *best* candidate, rather than being certain that our algorithm always produces the correct answer.

After experimenting with a few f_0 algorithms, we decided to use a modified YIN algorithm [10]. YIN operates in the time domain, relying on the assumption that $x_t - x_{t+\tau}$ is minimal when τ is the period of the signal. Finding the period is then a matter of picking the smallest $d_t(\tau)$ in (3.4), where W is the window size and x_j are the samples.

$$d_t(\tau) = \sum_{j=1}^{W} (x_j - x_{j+\tau})^2 \tag{3.4}$$

YIN contains a number of measures to increase the speed of this calculation and avoid errors, but a detailed examination of the algorithm lies outside the scope of this thesis.

Brossier modified the YIN algorithm to compute the square difference function $d_t(\tau)$ in the spectral domain [6] as well as a few extra measures to reduce errors, but a detailed examination of this also lies outside the scope of this thesis.

Using Brossier's spectral YIN algorithm as implemented in the aubio C library (`http://www.aubio.org`), with a window size of 1024 and hop size of 512, analyzing an amateur violinist performing Figure 3.5 produces Figure 3.6. Before using the modified YIN algorithm, we applied a silence gate: any frame which was too quiet received a value of 0. Since humans perceive pitches on a logarithmic scale and our ultimate goal is to grade intonation exercises, we convert all pitches into MIDI

note values.

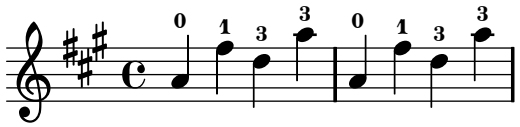


Figure 3.5: Sample violin intonation exercise.

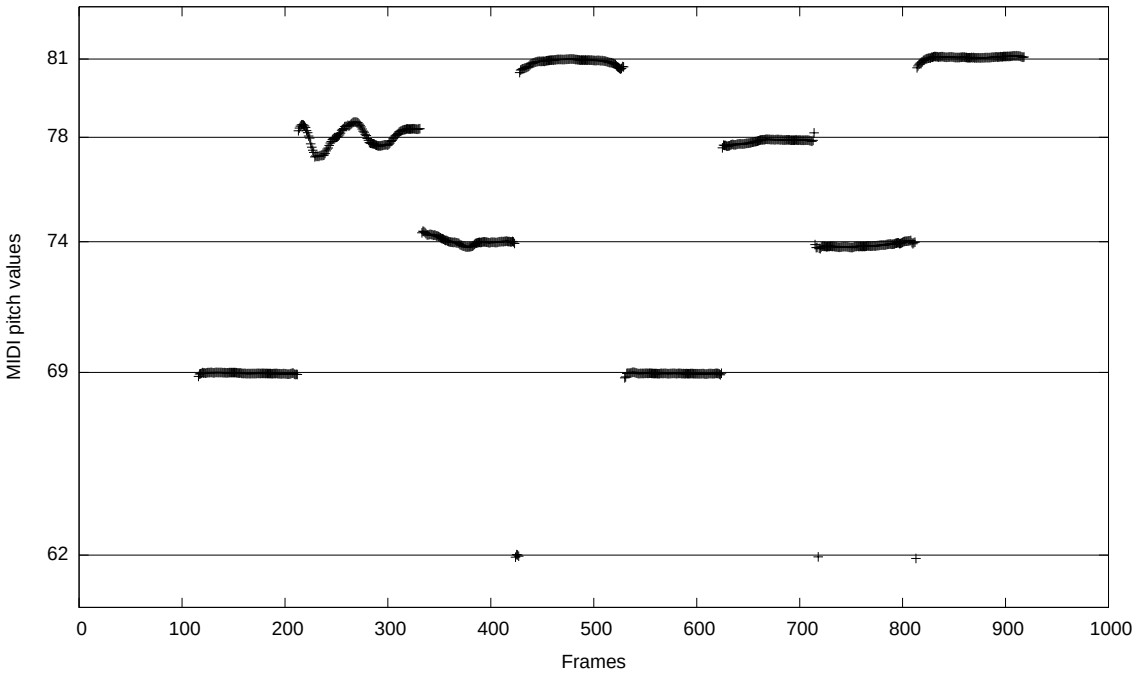


Figure 3.6: Pitches from a violin intonation exercise performance. The violinist did not prepare his hand position before beginning the exercise: the first note is an open string, while the second note was used to find the correct hand position. The subsequent notes were much more accurate.

3.3.2 Note Segmentation

Having determined the pitch of each frame, we group frames together to form notes. We can easily identify notes by glancing at Figure 3.6, but we need an automatic process to perform this segmentation.

Algorithm 2 performs this task adequately; Figure 3.7 shows the output of the algorithm. It scans through the pitches, finding the median value from a window of K_f pitch frames. If the current median is significantly different from the previous note, it marks the current frame as a note boundary. K_f was set to $34 \approx 0.4 \cdot \frac{44100}{512}$ (i.e. the minimum note length is 0.4 seconds), and K_p was set to 0.5.

Algorithm 2 Segment pitches into notes

Require: P // list of pitches
Require: K_f // constant: minimum number of frames in a note
Require: K_p // constant: minimum pitch change for a note boundary
  $B \leftarrow []$ // list of note boundaries
  $j \leftarrow 0$ // index for B
  $o_m \leftarrow 0$ // old (previous) median value of note boundary
  $o_i \leftarrow 0$ // old (previous) frame of note boundary
  for all i in length(P) **do**
    $m \leftarrow \text{median}(P_i, P_{i+1}, \ldots, P_{i+K_f})$
    if $(|m - o_m| > K_p)$ **then**
      if $(i > o_i + K_f)$ **then** // new boundary found
        $B_j \leftarrow i$
        $j \leftarrow j + 1$
        $o_m \leftarrow m$
        $o_i \leftarrow i$
      else // update previous boundary
        $B_j \leftarrow i$
        $o_m \leftarrow m$
        $o_i \leftarrow i$
      end if
    end if
  end for
  return B

3.3.3 Discussion of Intonation Transcription

Readers familiar with the current state of automatic music transcription may be surprised by the simplicity of our note segmentation. Algorithm 2 relies on three assumptions: that note boundaries have sharp changes in the pitch, that notes have a stable f_0 as detected by the modified YIN algorithm, and that all notes are longer than 0.4 seconds.

It may seem obvious that violin notes will contain sharp changes in pitch – putting down a new finger will abruptly change the pitch – but this assumption only holds for technical exercises. When violinists play real music, they occasionally add *glissandi* for expressive purposes. In contrast, when playing technical exercises, violinists try to keep each note as clear as possible.

Not all instruments have a stable f_0. Piano notes in particular are not stable – as the note decays, the difficulty of finding f_0 increases, resulting in unreliable values. Bowed violins produce very stable f_0 values.

The assumption that all notes are longer than 0.4 seconds will not apply to music

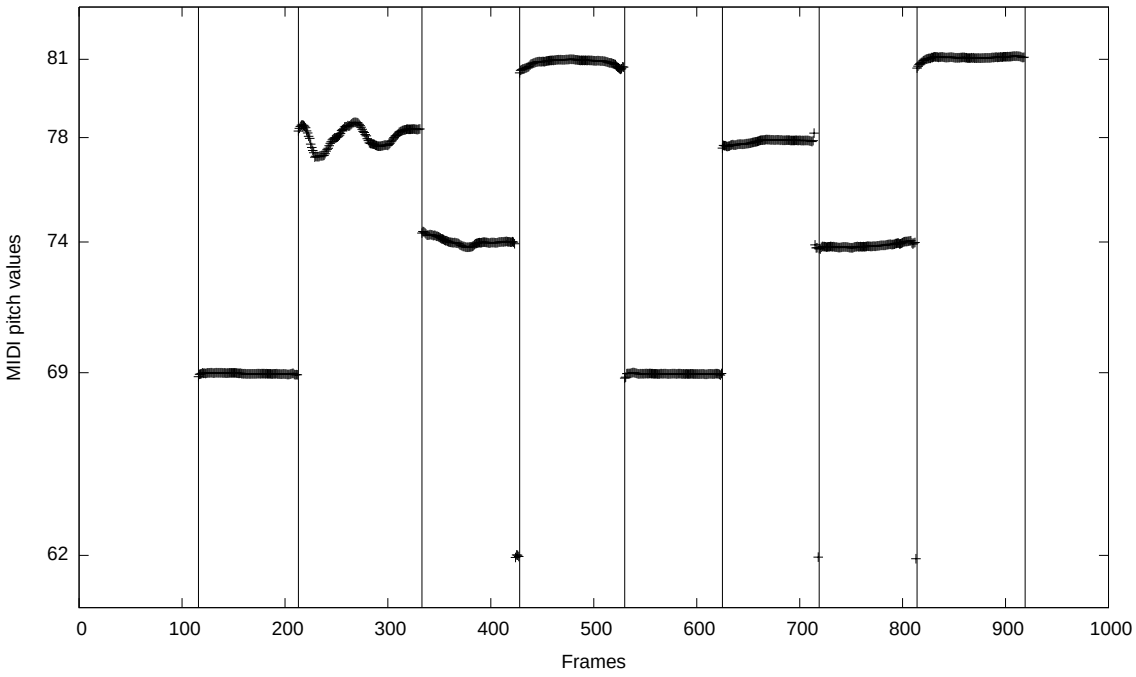


Figure 3.7: Segmented notes of a violin intonation exercise performance.

in general. For reference, this translates to eighth notes at 75 BPM. Most of our rhythmic clapping exercises contain notes faster than this, as does virtually any piece of real violin music.

3.4 Grading Intonation

When creating an algorithm for automatically grading intonation exercises, we consider the following goals:

1. The closer a student gets to the correct answer, the higher the grade should be. However, we should not expect students to play precisely the correct pitch.[4]

2. Grades should not vary based on the length of the notes in the exercise.

3. A student should fail the exercise if she plays a wrong note.

4. We should track the direction of the intonation error in addition to the magnitude.[5]

[4]Unlike clapping with a rhythmic accuracy of 12 milliseconds, playing a violin with perfect pitch *is* humanly impossible.

[5]Technically this is a requirement for visualization and not generating a grade, but it is convenient to discuss these calculations as part of the grading.

The intonation grading will also be based on MSE (3.5).

$$\text{Error} = \frac{1}{n} \cdot \sum_{i=1}^{n} (P_i - Q(i))^2 \tag{3.5}$$

where P_i are the detected pitches, and $Q(i)$ is the expected pitch for frame i. P is indexed starting from 1. When grading intonation exercises, we do not consider the rhythm, so $Q(i)$ is assigned based on the detected note boundaries and the exercise.

3.4.1 Calculating Intonation Errors

Although the modified YIN algorithm is very good at analyzing violin sounds, it still makes occasional mistakes. In Figure 3.6 and Figure 3.7 we see a few frames reporting a pitch of 62 instead of 74 (an octave error).[6] When grading performance, we should not penalize the student for errors in f_0 detection. To do this, we ignore any pitches that are more than K_r units from the median pitch of the note. We set $K_r = 2$ (two semitones). We also applied a maximum amount of error for each note; $K_o = 1$ (one semitone).

In addition to the magnitude of error, we also wish to track the direction; our visualization of the performance will show the user whether a note was too high, too low, or varying in both directions. Algorithm 3 shows how we calculate the magnitude and direction of error for a single note.

By default the expected pitch q is the MIDI note value. However, the user may choose to use just intonation[7] instead of equal temperament. In just intonation mode, the program takes the first note of the exercise as the tonic of the key. The expected pitch is calculated based on the scale degree of the expected note.

To get the overall score, we sum all the note errors, multiply by K_c, and subtract that from 100. After experimentation, we set $K_c = 50$. Applying this algorithm to the exercise in Figure 3.7 produces an overall grade of 86.9%.

[6] Octave errors are the most common error with f_0 detection algorithms: the detected frequency will be half or double the actual frequency. They are especially common at the beginnings and endings of notes.

[7] Under "equal temperament," the octave is divided into twelve equal semitones, where the frequency of each semitone is $\sqrt[12]{2} \approx 1.0595$ times the frequency of the previous semitone. Under "just intonation," simple ratios are used instead: A fifth is the ratio of 3 : 2, a fourth is 4 : 3, a major third is 5 : 4, and so on.

The biggest difference between the two temperaments occurs on the minor seventh of a scale. For example, in the key of A major or minor (taking $A = 440$ Hz), a G will be 783.99 Hz under equal temperament, and 770 Hz under just intonation. The difference is quite audible.

Algorithm 3 Calculating Intonation Errors

Require: P // list of detected pitches for the note
Require: q // the expected pitch of the note
Require: K_r // constant: allowable range of pitches from median
Require: K_o // constant: maximum error magnitude per note
  $e \leftarrow 0$ // magnitude of error
  $d \leftarrow 0$ // direction of error
  $c \leftarrow 0$ // count frames
  $m \leftarrow \text{median}(P_1, P_2, \ldots P_n)$
  for all i in length(P) **do**
    if ($|P_i - m| \leq K_r$) **then** // the pitch is close enough to the median
      $x \leftarrow (P_i - q)$
      $e \leftarrow e + x^2$
      $d \leftarrow d + x$
      $c \leftarrow c + 1$
    end if
  end for
  if ($c > 0$) **then** // normalize for note length
    $e \leftarrow e/c$
    $d \leftarrow d/c$
  end if
  // decide whether note is sharp, flat, or both
  if ($d < -0.1 \cdot \sqrt{m}$) **then**
    $d \leftarrow -1$
  else if ($d > 0.1 \cdot \sqrt{m}$) **then**
    $d \leftarrow 1$
  else
    $d \leftarrow 0$
  end if
  if ($m > K_o$) **then** // clip maximum error magnitude
    $m \leftarrow K_o$
  end if
  return e, d

3.4.2 Discussion of Intonation Grading

As with the rhythm grading, we have no hard evidence that MSE is an appropriate measure of intonation errors, nor do we have any hard evidence in favor of the constants chosen. Future work in this area would involve a study in which music teachers rate exercises with varying amounts of intonation errors.

One possible extension to our grading algorithm is to weigh the frames within each note differently. The ultimate goal of these intonation exercises is *not* to have the student playing with perfect intonation, but rather to develop the student's ability to

correct her own intonation. Improving the student's intonation is a secondary goal.

The distinction is important: no violinist plays perfectly in tune. Very skilled violinists will place their fingers *close* to the correct places, but they must still continually correct their finger positions. This functions much like our sense of balance: when sitting or walking, we do not consciously think about our balance, yet our brain is constantly directing our body to shift positions. We only consciously think about fixing our balance when performing difficult tasks such as riding a bicycle while holding a suitcase in one hand.

With this in mind, it would make sense to reward students who fixed their intonation in the middle of a note. This is already done to some extent, since the frames with fixed intonation would add much less to the error total. However, it may be desirable to apply additional weighting to the frames; something like a logarithmic function, where the weighing of the first frame is 0 and the weighting of the final frames approach 1. Again, a future survey of music teachers would be very valuable for judging whether this produces more accurate grades.

One potential problem with our intonation grading is that it does not allow violinists to play with vibrato. Skilled violinists normally add vibrato to notes; doing so in the context of these intontation exercises will result in a very poor grade. However, as long as the violinist is aware that she must not use vibrato, this problem may be avoided – violinists often practice scales without vibrato, so asking them to suppress their vibrato is quite reasonable.

3.5 Conclusion

In this chapter, we discussed the analysis of audio from student performances. From a list of approximately half a million samples of digitized audio, we calculated a list of approximately a dozen numbers, representing the musical events in the performance. These detected musical events were compared to the expected musical events, resulting in a list of errors. The sum of these errors produced a single score for the entire performance.

Chapter 4

Game Design and Visualization

This chapter brings together the exercises from Chapter 2 and the analysis of Chapter 3 in the form of a game. Students are presented with an exercise to perform with an optional metronome. After a student has attempted the exercise, her performance is graded and the results are presented visually.

In the "game" mode, a random exercise is chosen from each difficulty level. If the student receives a grade of 80% or higher, she has "passed" the exercise, and a new exercise is randomly chosen. When the student has passed two exercises on a level, she advances to the next level.

Exercises may be also chosen individually. In such cases, when a student has passed an exercise, she is prompted to choose another exercise.

4.1 Common Features

The audio from each performance is stored so that students may listen to it while examining the visualization of their performance. Each exercise may contain any number of "attempts", as shown in Figure 4.1. Each attempt is graded and stored individually.

If the performance analysis varies considerably from the expected values, the color background is changed. Red indicates that something is seriously wrong, while yellow indicates a warning that there was probably a mistake. These mistakes are generally either a problem with the transcription (i.e. the program fails to detect the claps or segment the notes), or the student misreading the exercise (i.e. playing the wrong notes). In these cases, there is little point in examining the exact score or output, as the analysis probably does not reflect the true ability of the student.

Both types of exercises have two levels of visualization: the main display, and the extra display. In both cases, the main display shows the detected errors and suggests possible improvements. The extra display, reachable by double-clicking on the main

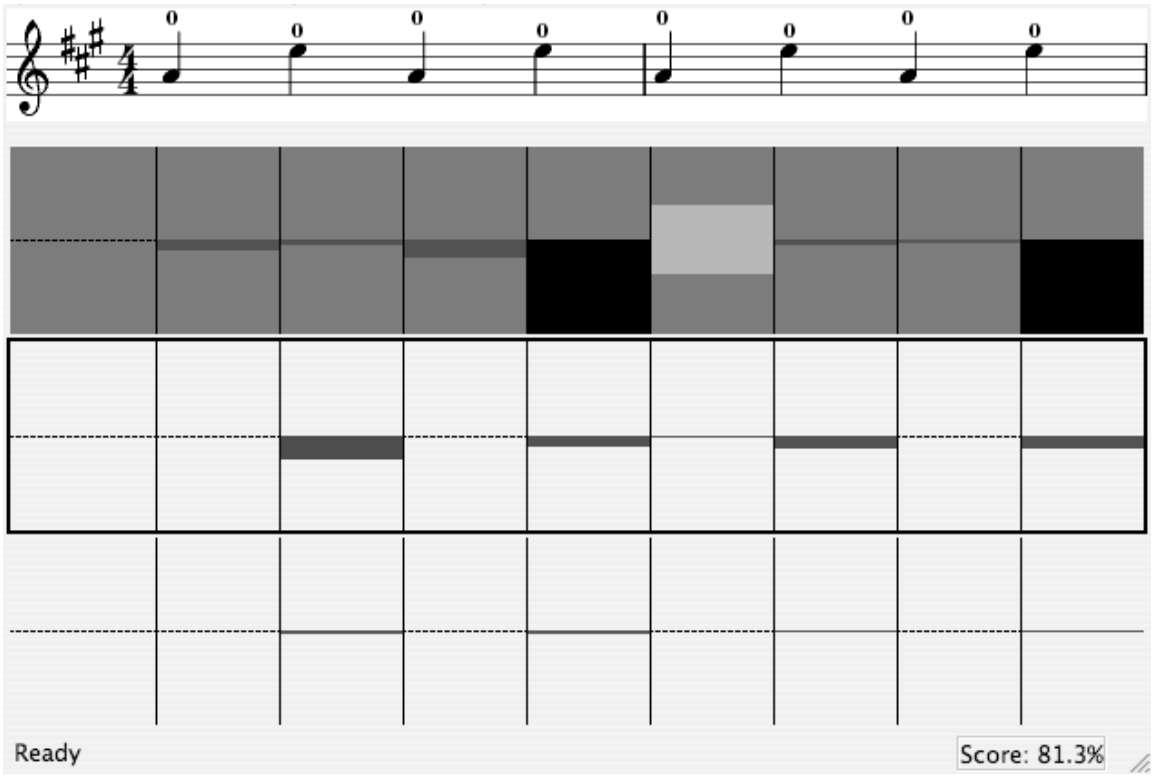


Figure 4.1: Intonation exercise with three attempts. The second attempt is selected; the grade in the bottom-right corner refers to this attempt. In the first attempt, the student played two Bs instead of Es; the shaded background indicates that performance was very different from the exercise. In the third attempt, the pitches were almost perfect.

display, shows low-level amplitude or pitch information. The extra display may be useful for students or teachers curious about transcription mechanism, but is not intended to be directly helpful for correcting student mistakes.

4.1.1 Levels of Supplied Metronome

Students may observe a metronome while recording or listening to an exercise. Different types of metronomes may be selected for recording and listening. There are four types of metronome:

1. Audible metronome ("rock musicians"): beats are indicated with audio (a drum), optionally with flashing light(s). The use of headphones is recommended. However, if the musician's claps are louder than the audio beats, the drum sounds will not interfere with the transcription of the musician's claps.

   This level is suitable for musicians who play in groups with constant percussion, such as rock musicians or instruments in many non-Western traditions.

2. Visual metronome ("classical musicians"): beats are only indicated with flashing light(s). Musicians may choose between a (small) flashing icon in a toolbar, an independent (potentially large) window containing a flashing circle, or an independent window containing two circles which blink on and off in alternation.

   This level is suitable for musicians who play in groups with conductors but without constant percussion, such as orchestral musicians.

3. Introduction only ("conductors"): the first four beats are indicated with a flashing light(s), and then no other beats are shown. Musicians must perform the exercise at the indicated tempo without any additional aid.

   This level is suitable for musicians who set the tempo for other musicians, such as conductors or percussionists.

4. None: there is no audible or visual metronome.

   This is suitable for listening to a previous attempt.

4.1.2 User-Created Exercises

In addition to the exercises created in Chapter 2, users may create their own exercises. This may be useful if they have difficulty with a particular passage in a piece of music. Students – or their teachers – may add a few bars of the music to this program, allowing the student to practice, receive objective analysis, and view the visualization of their errors.

Technical details of including user-created exercises (or even entirely new games) are presented in Section B.3.2.

4.2 Rhythm Games

Students are encouraged to use the metronome while performing rhythm exercises. Students may alter two aspects of the transcription and grading algorithms: the onset threshold K_t, and forcing the alignment to the first beat instead of the optimal alignment (3.3).

4.2.1 Displaying Rhythm Errors

When the performance has been analyzed, the main display shows the student the expected and detected claps. Figure 4.2 shows an example of this; it uses the same performance as was used in Figure 3.2a and Figure 3.4. To understand the display, users only need to know that time progresses from left to right, that the upper black

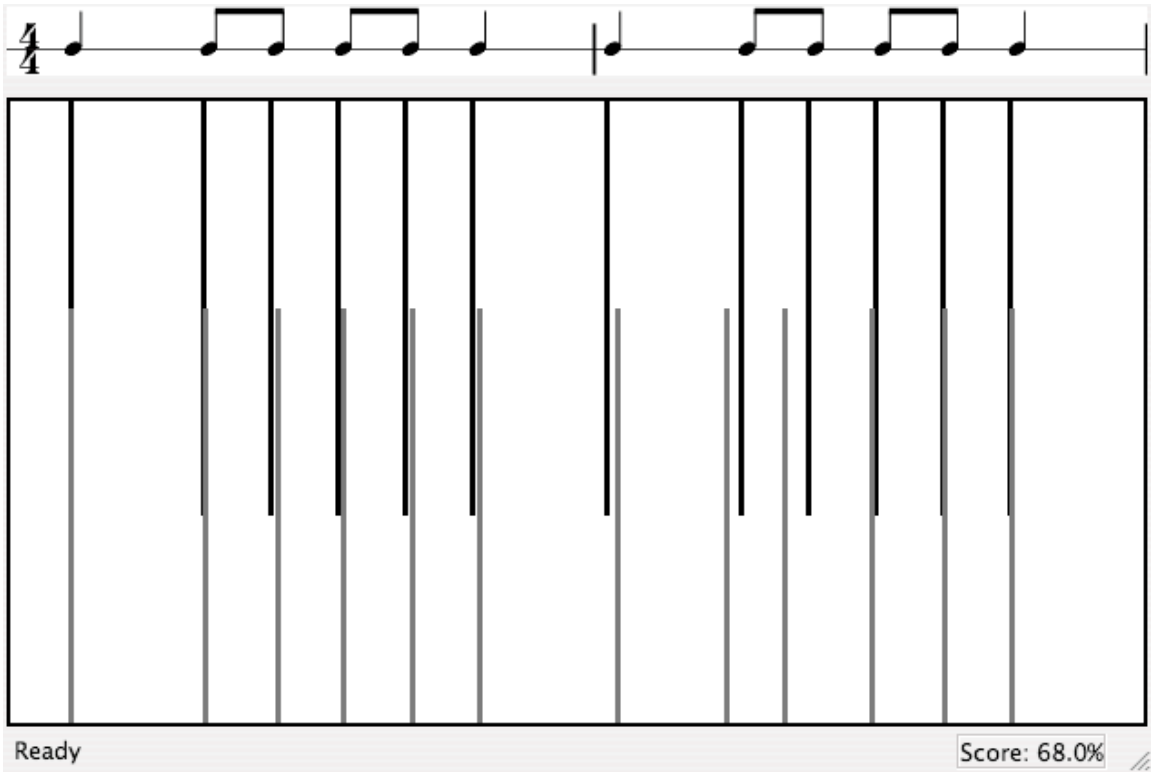


Figure 4.2: Rhythm performance (main display). The top black vertical lines show the expected claps, while the bottom gray (red in the program display) vertical lines indicate the detected claps. If the claps line up perfectly, there will be a single (multi-colored) line from the top to bottom of the display.

bars show where they *should* clap, and the lower gray bars (displayed in red in the actual program) show where they *did* clap.

The background of Figure 4.2 gives a warning based on the absolute difference between the number of detected claps and the expected claps. If the difference is 0, the background turns white (ok). If the difference is 1, 2, or 3, the background turns yellow (warning). If the difference is greater, the background turns red (serious problem).

Figure 4.3 shows the main display of an example with almost perfect rhythm, but an incorrect tempo. However, this visualization does not immediately indicate that the tempo was too fast; rather, it prompts a student to wonder why her first note was so late. In such cases, it may be preferable to align the first detected onset with the first expected onset, regardless of whether this produces an optimal score or not.

Extra information about the transcription is shown in Figure 4.4. Using this display, students in a particularly noisy room or with particularly inconsistent volume of claps may raise or lower K_t to optimize the clap detection.

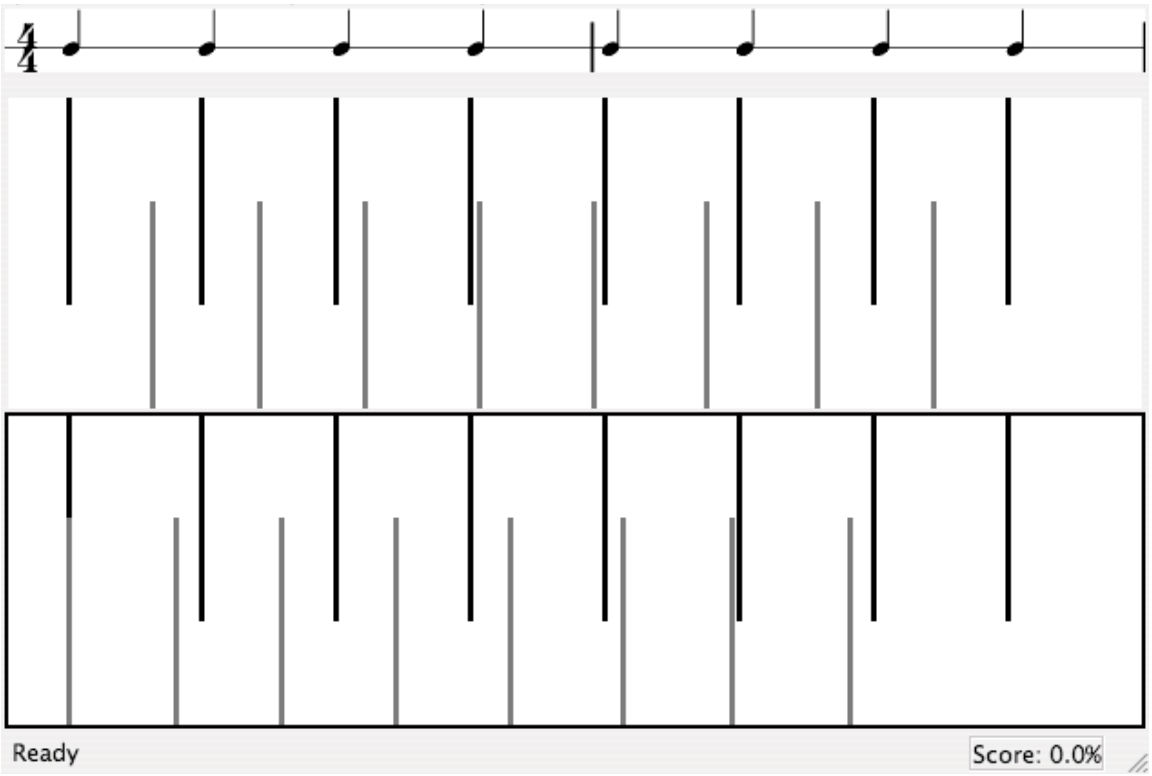


Figure 4.3: Rhythm performance (main display) with incorrect tempo. (top) Normal alignment. (bottom) first beat alignment.

4.2.2 Discussion of Rhythm Game

The rhythmic grading of Section 3.2 may be criticized for grading a musician's tempo stability, in addition to their ability to perform rhythms. Leaving aside the question of whether it actually makes sense to perform a "correct rhythm" if the tempo varies significantly within two bars, it is certainly true that our grading algorithm punishes students who do not follow the specified tempo. This is particularly apparent with the 0% grade in Figure 4.3. Could this be avoided?

A human listener – particularly a skilled musician – can easily infer an overall tempo from a single bar, and grade the rhythm accordingly. Automatic tempo detection is an ongoing area of research; computers are nowhere near as good as humans when it comes to "picking up" a tempo. Attempting to detect the tempo gives us two problems to solve simultaneously: pick the best tempo, and pick the best onset alignment. Solving these problems together is significantly more difficult than solving only one – especially since we cannot be certain that the student will perform the correct rhythm at all!

In addition, we should consider whether an externally-imposed tempo is actually a

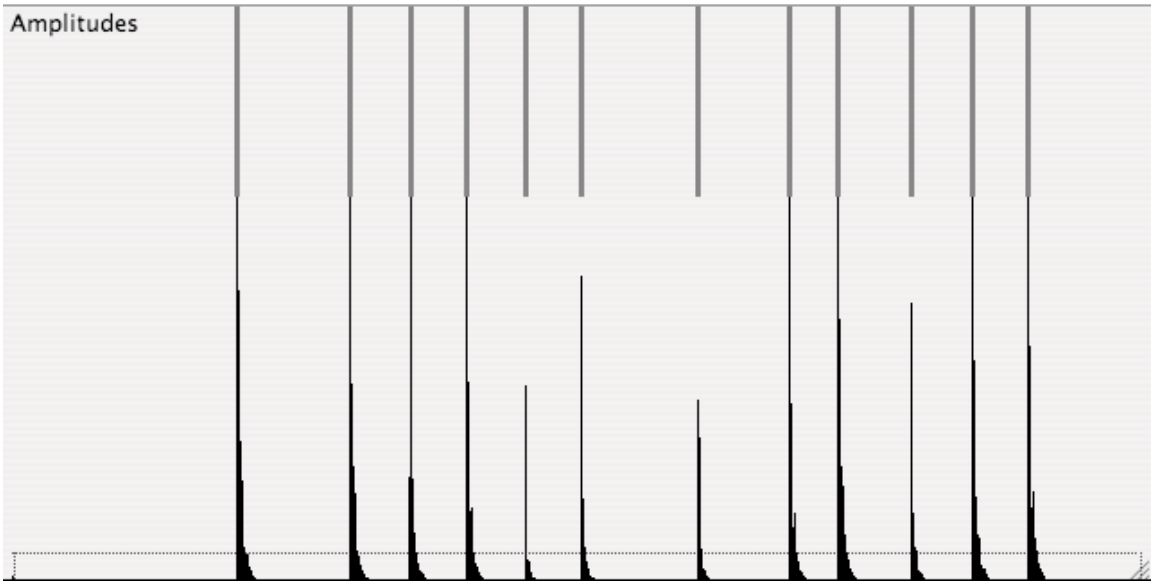


Figure 4.4: Rhythm performance (extra display). The RMS (amplitude) values extracted from the audio are shown in black lines on the bottom; the thick gray (red in the program display) lines on the top indicate detected onsets. The faint dotted horizontal line (blue in the program display) indicates K_t, the threshold value for peak detection.

problem. We argue that it is not: in almost every musical genre, when musicians play together, they play at the same tempo. Rock guitarists follow the drums. Orchestras follow the conductor. Chamber musicians negotiate the tempo amongst themselves and may exchange roles of setting tempo or following it – but all members must be able to follow the tempo set by other members.

Playing strictly according to a metronome is un-musical, but our exercises are not intended to be expressive music. Performing a solo Bach suite with strict tempo would be horrible, but clapping with a metronome for two bars would simply be good practice technique.

The rhythm game fulfills the overall goals discussed in Section 1.3.3:

Focus: Students must follow the music and metronome in addition to concentrating on their instrument. Since their instrument is simply their hands, students should have no difficulty performing the claps while reading the music and listening or watching the metronome. Since the metronome is configurable, students may choose whether they find the audio or visual metronome to be easier to follow, ensuring that they may concentrate on clapping the rhythm.

Reliability: The changing background color gives warnings about potential problems in the transcriptions, and these problems are easy to investigate (and

potentially fix by modifying K_t) by studying the extra display.

Transparency: As long as users can grasp the idea of digital sound being displayed as a set of amplitudes, the ensuing steps of recognizing onsets, finding differences between detected and expected onsets, and calculating a final grade are relatively simple. Grasping the connection between audio and a graphical display of amplitudes is not trivial, but given the stark contrast between the amplitudes of claps and background noise, most users should be comfortable with this step after a few experiments.

Simplicity: Clapping is one of the simplest forms of human music-making, as well as being one of the easiest events to detect in audio.

4.3 Violin Intonation Games

Students may alter one aspect of the transcription: K_f, the minimum note length.

4.3.1 Displaying Intonation Errors

The main violin intonation display is shown in Figure 4.5. This display should be understandable by students or parents with very little musical knowledge: if a note is blue, move the finger higher. If a note is purple, move the finger lower. If a note is green, stop moving the finger around in the middle of a note.

The background of Figure 4.5 gives a warning based on the number of notes whose pitch was at least one semitone away from the correct pitch. If there were 0 bad notes, the background turns white (ok). If there was 1 bad note, the background turns yellow (warning). If there were 2 or more bad notes, the background turns red (serious problem).

Figure 4.6 shows the extra display for violin intonation. This may be used to experiment with different values for K_f – if the value of K_f is too high or too low, the note boundaries will not match up with our intuitive sense of where the boundaries should be.

4.3.2 Discussion of Violin Intonation Game

The scope of the violin intonation game is much more limited than the rhythm game. The rhythmic exercises cover the first few years of rhythms that a classical violinist will see. The lack of alternative time signatures such as $\frac{3}{4}$ or $\frac{6}{8}$ is unfortunate, but these could be added without changing the overall approach.

In contrast, the violin intonation game only covers the first year of pitches that most students would encounter. The transcription is only suitable for relatively long

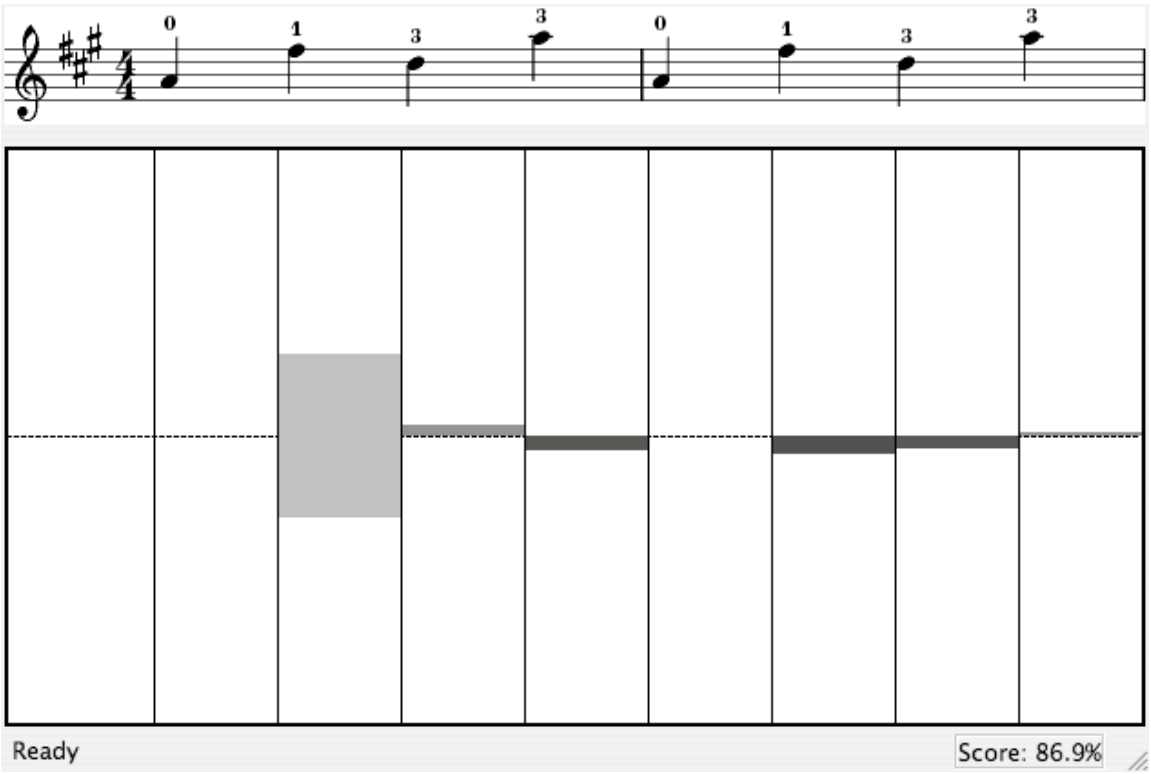


Figure 4.5: Intonation performance (main display). The intonation error for each note is shown. Bars above the central line (purple in the program display) indicate that the note was generally too sharp, while bars below the central line (blue in the program display) indicate that the note was generally too flat. Bars both above and below (green in the program display) bars indicate that the note alternated between too sharp and too flat. The color shade and the height of the box indicate the magnitude of the error.

notes. This is acceptable for slow "warm up" exercises, but it would be nice if the transcription could handle faster exercises as well.

Switching between equal temperament and just intonation is primarily a cosmetic feature. On most exercises, switching between the two temperaments only changes the final grade by a few percentage points. However, this feature may still be useful in blunting criticisms or distrust from students. If a violinist receives a bad score on the equal temperament scale, she may complain "oh, violinists don't use the out-of-tune 'piano tuning' when we play by ourselves. What silly software! I should have gotten a higher grade!". With the ability to switch to just intonation, such a violinist's claim may be tested.

The transfer of exercise data between creation and analysis is not ideal – the key is not explicitly stated. Due to this fact, the exercise must start with the tonic, and the analysis always treats the first note as such for the just intonation grading mode.

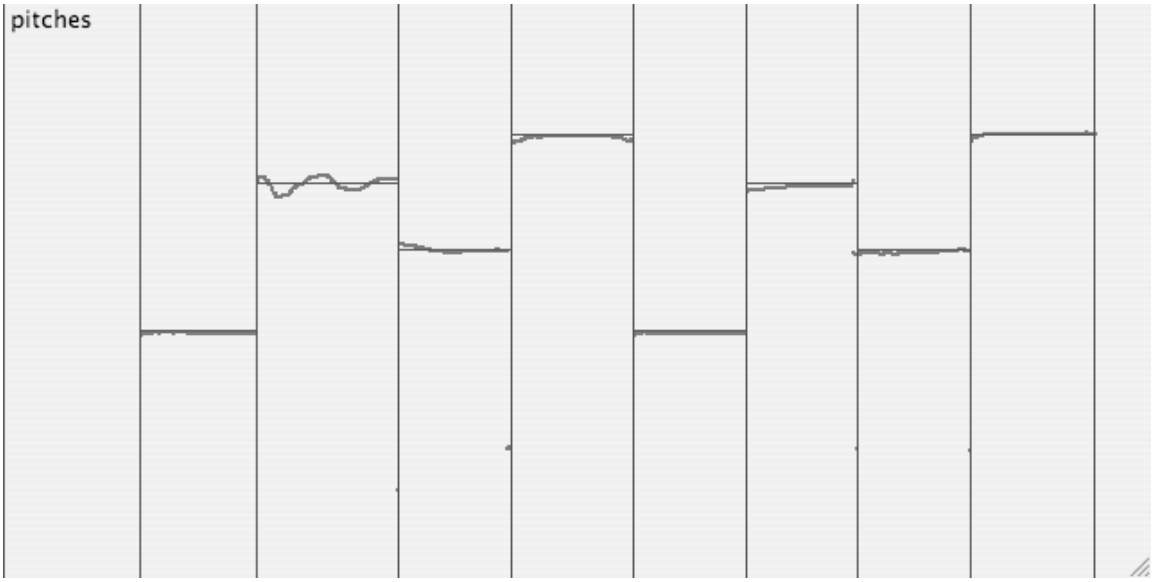


Figure 4.6: Intonation performance (extra display). The gray dots (red in the program display) indicate the pitch of each frame. The vertical lines (blue in the program display) indicate the note boundaries, while the horizontal lines (also blue in the program display) indicate the expected pitch of each note.

This limits the number of automatically created exercises, which may or may not be a bad thing, since it reduces the size of downloaded files. However, it may pose a problem for user-created intonation exercises.

The violin intonation game mostly fulfills the overall goals discussed in Section 1.3.3:

Focus: Students must follow the music, follow the metronome, and perform the exercise on their instrument. Reading music and playing a violin at the same time should be nothing new to music students with a few weeks' experience. For some students, a metronome may be an unusual distraction, but this is unavoidable.

Reliability: If a problem arises in the transcription, there will probably be at least one note which has a large absolute error. This triggers a warning in the form of the background color. Such problems are easily investigated (and possibly fixed by modifying K_m) by examining the extra display.

Transparency: The intonation game is as transparent as possible. The initial pitch detection is not transparent – unless users look at [10] and [6] themselves, they will not know how the program generates the list of pitches. That said, in some

ways the connection between audio and the graphical pitch display is easier than the connection between audio and the graphical amplitude display. Humans – especially music students – naturally think about sound being composed of pitches; unlearning this notion is a non-trivial task for students studying digital audio. In any case, most users should be able to recognize the connection between audio and the graphical pitch display after a few experiments.

Once the users trust the graphical pitch display, the rest is relatively easy to follow. Segmenting pitches into notes is easily understood, especially if the users experiment with different values for the minimum note length. Comparing the actual pitches with the expected pitches can be done by eye, and noting the relationship between the size of the error bars and the final score is trivial.

Simplicity: The intonation game is relatively simple, but a number of trade-offs were made.

Users must choose whether to use equal temperament or just intonation. This introduces a choice which may confuse many users; even some university music students do not realize that there is more than one tuning system. We certainly cannot assume that beginning violin students are familiar with the distinction! We recommend that music teachers assign one system or the other, and instruct the students not to modify this setting.

By using a large value for the minimum note length, we achieved good results with a simple note segmentation algorithm. As a result, the grading process is much easier to understand than it might otherwise be. However, 0.4 seconds is quite long for violin technical exercises. In real music, violins often play shorter notes; many technical exercises for violin explicitly test playing quickly. In this case, our attempt at simplicity has compromised the usefulness of the CAMIT program.

One possible simplification would be to accept both equal temperament and just intonation. The grading algorithm could grade the exercise according to both systems, and pick the best grade. Alternatively, the grading system could switch tuning systems for each note. This question should be investigated as part of a survey of music teachers. Would music teachers rank performances which switched between tuning systems highly? Would music teachers rank performances which played "in between" the two systems highly?

One additional feature which may be useful would be to have the computer play back the performance with perfect intonation, but still using the student's sound. This is possible by performing pitch shifting as described in [22]. The benefit of using the student's own sound is that the student hears what she should actually be playing. If the "perfect version" is performed with MIDI sounds, the result sounds nothing like a real violin. If the "perfect version" is simply a recording of a professional musician playing an expensive violin, the student may justifiably point out that it would be impossible for her to sound exactly like that.

4.4 Conclusion

In this chapter, we discussed our method of visualizing student errors in a novel display, understandable even by non-musicians. We discussed the ways in which users may customize the software, both to produce more accurate analysis results, and to create entirely new exercises to practice.

We also examined the overall structure of the games: are they actually testing the material we wish to test? Will the grade reflect the actual abilities of students? Do these games fulfill our overall goals of CAMIT projects? Due to the very focused nature of the technical exercises, we can answer all these questions in the affirmative.

Chapter 5

Conclusion and Future Work

This research developed a Computer-Assisted Musical Instrument Tutoring system to help music students. The rhythmic exercises will help all students; greater familiarity with rhythms allows the student to concentrate on other aspects of their playing. The violin intonation exercises are targeted at violin students in their first year of study.

In order to accomplish this goal, we discussed the general field of CAMIT projects and identified specific goals to improve such projects. We showed how these goals could be applied to this research, and how they were fulfilled in MEAWS.

We began by investigating the automatic creation of exercises for students. This is a new area, and our attempts produced mixed results. Crafting difficulty levels and creating all exercises on each level works well for small, targeted situations, but this approach will not scale up to larger, more general projects. Future work on automatic exercise creation should involve exercises being created in near-realtime, ideally during the practice session itself.

Crafting targeted exercises allowed us to use simple transcription algorithms without sacrificing accuracy. As the field of music transcription matures, it may be appropriate to relax some of the constraints on exercise generation. However, in some cases the constraints may be musically appropriate: forcing the pitch to change for each note is quite reasonable for practicing intonation.

5.1 Future Work

This thesis has identified a number of important areas for future research. One of the most interesting ideas is testing the grading scheme against the judgments of experienced music teachers.

5.1.1 Objective Grading of Technical Skills

An objective, widely-accepted method of grading basic technical skills like rhythmic and intonation accuracy would be a great boon to music education. In addition to

mundane uses such as as grading student exercise performances (as in MEAWS), this would provide real quantitative data about debates in music education.

For example, one unanswered question for violin education is whether "fingerboard tape" is a good idea. "Fingerboard tapes" are small pieces of tape – thinly-cut electrical tape, masking tape, or stickers – which are placed on the violin fingerboard to show where the left-hand fingers should be placed. Proponents suggest that the tape helps students learn the correct hand position; after six months to a year, the tape is removed. Opponents argue that the tape hinders the development of a student's feedback loop. Skilled violinists must continually correct their finger positions based on the auditory output, so beginners should begin learning how to do this from the very first lesson. If students are shown the correct positions with tape, they argue, they will rely on the tape instead of their ears. This debate has lasted for decades (if not centuries!) and shows no sign of ending. Famous violinists and violin teachers argue on both sides of the issue.

However, the debate could be settled if we could develop an objective intonation grading algorithm. We could gather a large number of new violin students – there are thousands of new violin students in music academies each year. Half of these students would be given tape, and half of them would not have tape. We would then test their intonation ability after six months, one year, two years, and five years. If the survey results showed that tape or no tape produced more accurate intonation, the question would be largely settled. If the survey results showed no statistically significant difference, it would be even more interesting – it would mean that it did not matter whether tape was used or not!

Objective testing would be useful for evaluating other questions. For example, when should violin students begin shifting? Some teachers prefer to leave the students in first position until they are very comfortable; this helps reinforce the initial hand position. Other teachers prefer to introduce shifting quite early; violinists should be comfortable in any position. Which method, if any, produces the best short-term and long-term results?

5.1.2 Attributing Cause to Errors

Another possibility for future work is in estimating the cause of errors, or in other words, splitting an error into its contributing factors. For example, if one particular note is 0.6 MIDI pitch values sharp, the software would determine that 0.3 of the error was caused from using the 4^{th} finger, 0.2 of the error was caused from playing in second position, and 0.1 of the error was random chance.

This kind of analysis would require a great deal of data from the student's previous performances in order to perform the data mining. However, if the student had practiced with this program for fifteen minutes each day for the past three months, it may be possible. The software would track the errors produced by each finger, each string, and each position. Using machine learning, it would break down each note's error into a combination of separate error factors.

5.1.3 Generating Exercises Just In Time

The ability to generate exercises in the middle of a practice session does not require any new research; the tools already exist. However, putting the tools together, in an easily-distributable format, is a non-trivial task.

On a GNU/Linux system, this can be done by stringing together programs on the command-line. Oz provides the exercise creation, LilyPond creates the `.png`, python creates the `.exp`, and MEAWS (or its successor) displays the image and grades the result. This system requires that the user install all this software.[1]

However, a GNU/Linux-only solution would vastly reduce the number of potential users. A solution for MacOS X and Windows would be more complicated, but must be found if this tool is to be widely used.

If an internet connection could be assumed, a thin client approach may work quite well: MEAWS (or its successor) could record and grade audio, and then communicate with a central server to generate the next exercise.

5.1.4 Constantly Modifying the Set of Constraints

If the previous two steps were completed, we could create truly targeted exercises. Once the software identifies a particular weakness, it could specifically include those features in future exercises. As the student continues to practice with the program, she spends all her time and effort developing the skills in greatest need of practice.

The program need not consider all skills evenly – if the student's teacher specifically wanted her to focus on rhythm for one week (or month), the teacher may configure the program to weigh rhythmic mistakes more heavily than others. Conversely, if the teacher feels that the student would benefit from working on other musical skills, the teacher may reduce the weighting of rhythmic mistakes. Such a system to generate targeted exercises *must* allow a human teacher to direct the overall

[1] On a Debian GNU/Linux system, this can be done with a one-line command:

```
sudo apt-get install mozart mozart-stdlib lilypond python
```

Other distributions include similar capabilities.

flow: as we stated in Chapter 1, no computer can ever replace a skilled teacher's estimation of a child's mood and personality. Any broadly-focused educational system which does not consider those human elements is doomed to failure.

Bibliography

[1] T. Anders. *Composing Music by Composing Rules: Design and Usage of a Generic Music Constraint System*. PhD thesis, School of Music & Sonic Arts, Queen's University Belfast, 2007.

[2] G. Assayag, C. Rueda, M. Laurson, C. Agon, and O. Delerue. Computer-Assisted Composition at IRCAM: From PatchWork to OpenMusic. *Computer Music Journal*, 23(3):59–72, 1999.

[3] J. P. Bello, L. Daudet, S. Abdallah, C. Duxbury, M. Davies, and M. B. Sandler. A Tutorial on Onset Detection in Music Signals. *IEEE Transactions on Speech and Audio Processing*, 13:1035–1047, Sept. 2005.

[4] J. Bilmes. Timing is of the Essence: Perceptual and Computational Techniques for Representing, Learning, and Reproducing Expressive Timing in Percussive Rhythm. Master's thesis, Massachusetts Institute of Technology, 1993.

[5] M. Brandao, G. Wiggins, and H. Pain. Computers in music education. In *Artificial Intelligence and the Simulation of Behavior '99 Symposium on Musical Creativity*, 1999.

[6] P. Brossier. *Automatic Annotation of Musical Audio for Interactive Applications*. PhD thesis, Queen Mary University of London, UK, August 2006.

[7] N. Collins. Using a pitch detector for onset detection. In *ISMIR: International Conference on Music Information Retrieval*, pages 100–106, 2005.

[8] R. B. Dannenberg, M. Sanchez, A. Joseph, R. Joseph, R. Saul, and P. Capell. A Computer-Based Multi-Media Tutor for Beginning Piano Students. *Journal of New Music Research*, 19(2-3):155–173, 1990.

[9] R. B. Dannenberg, M. Sanchez, A. Joseph, R. Joseph, R. Saul, and P. Capell. Results from the Piano Tutor Project. In *Proceedings of the Fourth Biennial Arts and Technology Symposium*, pages 143–150, 1993.

[10] A. de Cheveigne and H. Kawahara. Yin, a fundamental frequency estimator for speech and music. *The Journal of the Acoustical Society of America*, 111(4):1917–1930, 2002.

[11] S. Ferguson. Learning Musical Instrument Skills Through Interactive Sonification. In N. Schnell, F. Bevilacqua, M. J. Lyons, and A. Tanaka, editors, *New Interfaces for Musical Expression*, pages 384–389. IRCAM - Centre Pompidou in collaboration with Sorbonne University, 2006.

[12] S. Ferguson, A. V. Moere, and D. Cabrera. Seeing Sound: Real-Time Sound Visualisation in Visual Feedback Loops Used for Training Musicians. In *IEEE Proceedings of the International Conference on Information Visualisation*, pages 97–102, London, 2005.

[13] D. Fober, S. Letz, and Y. Orlarey. VEMUS - Une école de musique européenne virtuelle. In *Proceedings of the Journées d'Informatique Musicale*, pages 57–63, Lyon, France, 2007. in french.

[14] D. Fober, S. Letz, Y. Orlarey, A. Askenfeld, K. Hansen, and E. Schoonderwaldt. IMUTUS – an Interactive Music Tuition System. In *Proceedings of the Sound and Music Computing conference (SMC)*, pages 97–103, Paris, 2004.

[15] Free Software Foundation. GNU General Public License version 3. `http://www.gnu.org/licenses/gpl.html`.

[16] D. Hoppe, M. Sadakata, and P. Desain. Development of real-time visual feedback assistance in singing training: a review. *Journal of Computer Assisted Learning*, 22:308–316(9), August 2006.

[17] ICSRiM - University of Leeds. i-Maestro. http://www.i-maestro.net/, 2007.

[18] Institute for Language and Speech Processing, Greece. VEMUS: Virtual European Music School. http://www.vemus.org/, 2007.

[19] W. Jie, J. Boo, Y. Wang, and A. Loscos. A Violin Music Transcriber for Personalized Learning. In *Multimedia and Expo, 2006 IEEE International Conference on*, pages 2081–2084, Toronto, ON, Canada,, July 2006.

[20] D. Johnson. Project LRNJ: Learn Japanese RPG. http://lrnj.com, 2008.

[21] A. Klapuri and M. Davy, editors. *Signal Processing Methods for Music Transcription*. Springer, New York, 2006.

[22] M. Lagrange, G. Percival, and G. Tzanetakis. Adaptive harmonization and pitch correction of polyphonic audio using spectral clustering. In *Proceedings of the International Conference on Digital Audio Effects*, Bordeaux, France, September 2007.

[23] M. Laurson and M. Kuuskankare. A constraint based approach to musical textures and instrumental writing. In *Seventh International Conference on Principles and Practice of Constraint Programming, workshop on Musical Constraints*, Cyprus, 2001.

[24] D. B. Loeffler. Instrument Timbres and Pitch Estimation in Polyphonic Music. Master's thesis, Georgia Institute of Technology, April 2006.

[25] A. Loscos, Y. Wang, and W. J. J. Boo. Low Level Descriptors for Automatic Violin Transcription. In *ISMIR: International Conference on Music Information Retrieval*, pages 164–167, 2006.

[26] W. C. M. and Z. R. J. Influence of Tonal Context and Timbral Variation on Perception of Pitch. *Perception & Psychophysics*, 64:198–207(10), 1 February 2002.

[27] H. Nienhuys, J. Nieuwenhuizen, and G. Percival. Lilypond music notation. `http://www.lilypond.org/`, Feb. 2008.

[28] B. Ong, K. Ng, N. Mitolo, and P. Nesi. i-Maestro: Interactive multimedia environments for music education. In *i-Maestro 2nd Workshop on Technology Enhanced Music Education*, ICSRiM - University of Leeds, 2006.

[29] C. Oshima, K. Nishimoto, and M. Suzuki. Family Ensemble: a Collaborative Musical Edutainment System for Children and Parents. In H. Schulzrinne, N. Dimitrova, A. Sasse, S. B. Moon, and R. Lienhart, editors, *ACM Multimedia*, pages 556–563. ACM, 2004.

[30] G. Percival, T. Anders, and G. Tzanetakis. Generating targeted rhythmic exercises for music students with constraint satisfaction programming. In *Proceedings of the 2008 International Computer Music Conference*, Belfast, UK, August 2008.

[31] G. Percival, Y. Wang, and G. Tzanetakis. Effective Use of Multimedia for Computer-Assisted Musical Instrument Tutoring. In *EMME '07: Proceedings of*

the International Workshop on Educational Multimedia and Multimedia Education, pages 67–76, New York, NY, USA, 2007. ACM Press.

[32] M. Robine, G. Percival, and M. Lagrange. Analysis of saxophone performance for computer-assisted tutoring. In *Proceedings of the International Computer Music Conference (ICMC07)*, volume 2, pages 381–384, Copenhagen, Denmark, aug 2007.

[33] P. V. Roy and S. Haridi. *Concepts, Techniques, and Models of Computer Programming*. The MIT Press, March 2004.

[34] O. Sandred. Searching for a rhythmical language. In *PRIMSA (Pedagogia e Ricerca Internazionale sui Sistemi Musicali Assistiti)*. EuresisEdizioni, Milano, 2003.

[35] W. A. Schloss. *On the Automatic Transcription of Percussive Music: From Acoustic Signal to High Level Analysis*. PhD thesis, Stanford University, CA, USA, May 1985.

[36] E. Schoonderwaldt, A. Askenfeld, and K. Hansen. Design and Implementation of Automatic Evaluation of Recorder Performance in IMUTUS. In *Proceedings of the International Computer Music Conference (ICMC)*, pages 97–103, Barcelona, Spain, 2005.

[37] E. Schoonderwaldt, K. Hansen, and A. Askenfeld. IMUTUS – an Interactive System for Learning to Play a Musical Instrument. In *Proceedings of the International Conference of Interactive Computer Aided Learning (ICL)*, pages 143–150, Villach, Austria, 2004.

[38] T. Shiratori, A. Nakazawa, and K. Ikeuchi. Synthesizing dance performance using musical and motion features. *2006. ICRA 2006. Proceedings 2006 IEEE International Conference on Robotics and Automation*, pages 3654–3659, May 2006.

[39] S. W. Smoliar, J. A. Waterworth, and P. R. Kellock. pianoFORTE: a System for Piano Education beyond Notation Literacy. In *MULTIMEDIA '95: Proceedings of the third ACM international conference on Multimedia*, pages 457–465, New York, NY, USA, 1995. ACM Press.

[40] C. Truchet, G. Assayag, and P. Codognet. OMClouds, a heuristic solver for musical constraints. In *MIC2003: The Fifth Metaheuristics International Conference*, 2003.

[41] E. Tsang. *Foundations of Constraint Satisfaction*. Academic Press, London and San Diego, 1993. Out of print; available for download at `http://www.bracil.net/edward/FCS.html`.

[42] Y. Wang, B. Zhang, and O. Schleusing. Educational Violin Transcription by Fusing Multimedia Streams. In G. Friedland, W. Hürst, L. Knipping, and M. Mühlhäuser, editors, *ACM Multimedia EMME (Educational Multimedia and Multimedia Education) Workshop*, pages 57–66. ACM, 2007.

[43] Wikipedia. Guitar hero (series) — wikipedia, the free encyclopedia, 2008. [Online; accessed 10-March-2008].

[44] Wikipedia. Loom (video game) — wikipedia, the free encyclopedia, 2008. [Online; accessed 10-March-2008].

[45] Wikipedia. Rock band (video game) — wikipedia, the free encyclopedia, 2008. [Online; accessed 10-March-2008].

[46] M. Wright. *The Shape Of an Instant: Measuring and Modeling Perceptual Attack Time with Probability Density Functions*. PhD thesis, CCRMA, Music Dept., StanfordUniversity., 2008.

[47] J. Yin, Y. Wang, and D. Hsu. Digital Violin Tutor: an Integrated System for Beginning Violin Learners. In *MULTIMEDIA '05: Proceedings of the 13th annual ACM international conference on Multimedia*, pages 976–985, New York, NY, USA, 2005. ACM Press.

Appendix A

Papers by the Author

Portions of this thesis have appeared in previous publications; they are listed here in chronological order.

- **Adaptive Harmonization and Pitch Correction of Polyphonic Audio Using Spectral Clustering**, by Mathieu Lagrange, Graham Percival, and George Tzanetakis [22].

  This paper presented a method of splitting polyphonic audio (such as a violin with piano). This allows us to correct violin pitches without modifying the background sound, and provided the inspiration for some future work noted in Chapter 4.

- **Analysis of Saxophone Performance for Computer-Assisted Tutoring**, by Matthias Robine, Graham Percival, and Mathieu Lagrange [32].

  This was the first version of MEAWS, prompting much discussion about CAMIT projects and the software design in Appendix B.

- **Effective Use of Multimedia for Computer-Assisted Musical Instrument Tutoring**, by Graham Percival, Ye Wang, and George Tzanetakis [31].

  This paper provided the basis for Chapter 1.

- **Generating targeted rhythmic exercises for music students with constraint satisfaction programming**, by Graham Percival, Torsten Anders, and George Tzanetakis [30].

  This paper appears almost unchanged in the first half of Chapter 2.

Appendix B

Program Design and Source Code

The ideas in this research were implemented in MEAWS: Musician Evaluation and AWdition for Strings.

B.1 General Principles

The primary motivation in writing MEAWS was to create a useful tool for students and teachers. The second motivation was to test the ideas developed in this research. These two motivations lead to the following principles:

Cross-platform: In order for the software to be used as widely as possible, it should work on multiple operating systems.

Freely available: In order for the software to be used as widely as possible, it should be available under a copyright license which permits unlimited distribution. This allows teachers and students to share copies of this software with their friends.

Viewable source: Users should be able to view the source code. This is an important part of the overall goal of transparency, as discussed in Section 1.3.3. In addition, more scrutiny of the source code may result in bug fixes or suggestions for improvement.

Don't re-invent the wheel: Use existing libraries whenever possible. Rewriting code that is already available is both a waste of time and invites bugs, as existing software libraries will have received widespread testing.

Given these principles and the enormous quantity of existing software libraries available under the GPL, it makes sense to publish MEAWS under the GPL. The GPL [15] is a software license which essentially states "I'll share this software with

you, as long as you share any software that you build using it." The actual license is much more nuanced; do not take this one-sentence summary as a complete description of the license.

B.2 MEAWS implementation

MEAWS was written in C++. It was written with cross-platform portability in mind, and all libraries are cross-platform[1].

B.2.1 Libraries Used

MEAWS makes use of the following software libraries:

RtAudio provides a cross-platform Application Programming Interface (API) for realtime audio input and output.

libsndfile provides an API for reading and writing sound files.

aubio includes the modified YIN F_0 detection; it was created by Brossier [6].

Qt is a cross-platform framework for desktop applications. It includes creating GUIs, file handling, and a host of other functions for interacting with operating systems without requiring any rewriting of source code.

B.2.2 Classes

Figure B.1 shows the main classes in Meaws.

- MainWindow: this class handles the visual aspects of the main window and passes messages amongst its children.
- AudioBackend: this class interfaces with RtAudio and libsndfile, providing audio input/output with the computer microphone, speakers, and sound files.
- Metro: this classes handle the metronome. Large visual metronomes are created with the BigMetro class in a separate Qt window, while the audio metronome is performed with Qt's Sound class.
- User: this class contains persistent data about the current user. It stores user-supplied information such as their main instrument and years playing, in addition to MEAWS options such as the clapping threshold K_t and minimum note length.

[1] MEAWS has only been compiled and tested on MacOS X and Linux, but anybody familiar with Windows development should have no difficulty compiling it under Windows.

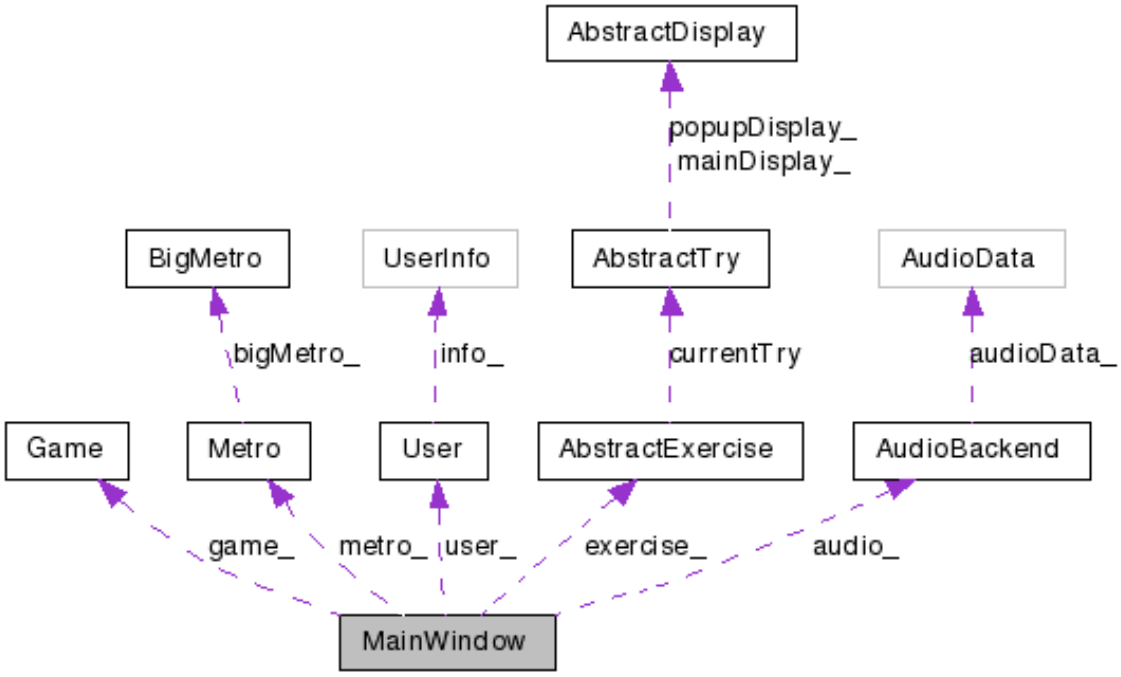


Figure B.1: Classes in MEAWS. Boxes with solid borders are classes, while boxes with faint borders are data structures. Text beside arrows are variable names.

- Game: this class handles game-related logic, such as reading game files from disk, picking an appropriate exercise to load, and evaluating whether the user should advance to the next level.

In addition to the above list, there are three abstract classes which are inherited by four classes for each level:

- AbstractExercise (inherited by RhythmExercise and IntonationExercise): this class displays the exercise and maintains a list of Try objects. It will always contain at least one Try, and may contain multiple Trys.

- AbstractTry (inherited by RhythmTry and IntonationTry): this class receives audio data from AudioBackend and analyzes the data (with Aubio in the case of IntontionTry) according to Chapter 3.

- AbstractDisplay (inherited by RhythmDisplayLines, RhythmDisplayAmps, IntonationDisplayBars, and IntonationDisplayPitches): this draws the result of the analysis according to Section 4.2.1 and Section 4.3.1. The mainDisplay_ objects are RhythmDisplayLines or IntonationDisplayBars, while the popupDisplay_ objects are RhythmDisplayAmps or IntonationDisplayPitches.

B.3 Creating Exercises in Oz and Strasheela

The overall music representation – linking the Event and Durations lists, or the String, Hand position, Finger, and Pitch lists – was added to Strasheela [1]. Common constraints such as "list must contain at least x values greater than y" were also added to Strasheela. They may be found in `contributions/gperciva/{eventdurs, rhythm-funcs, pitchfingers, intonation-funcs}`. Source code to produce individual levels was kept in a subdirectory of the MEAWS source code.

B.3.1 Sample: Functions Involved in Rhythms Level 2

As an example of constraint programming, we examine rhythm exercise level 2 in detail. This level introduced sixteenth notes in pairs replacing an eighth note. Relevant functions added to Strasheela:

```
proc {Setup Beats BeatDivisions Events Durations}
   NumEvents = Beats*BeatDivisions
in
   %% setup lists, add fake note at end
   Durations = {FD.list NumEvents 0#NumEvents}
   Events = {FD.list NumEvents+1 0#2}
   %% "courtesy one" to simply Duration defintions
   {Nth Events NumEvents+1} =: 1
   %% first onset can't be "continue note"
   {Nth Events 1} \=: 0

   for X in 1..NumEvents do
      %% setup maxmium durations
      {Nth Durations X} =<: NumEvents+1-X

      %% align rests
      {FD.equi
       ({Nth Durations X} =: 0)
       ({Nth Events X} =: 0)
       1}

      %% align 1 dur
      {FD.equi
       {Nth Durations X} =: 1
       {FD.conj
```

```
         ({Nth Events X} >: 0)
         ({Nth Events X+1} >: 0)
        }
        1}

      %% align 2+ dur
      for D in 2..NumEvents do
         if ( X =< (NumEvents+1-D) ) then
            {FD.equi
             ({Nth Durations X} =: D)
             {FD.conj
              ({Nth Events X} >: 0)
              {FoldR
               {LUtils.sublist Events X+1 X+D-1}
               fun {$ X Y}
                  {FD.conj (X=:0) Y}
               end
               ({Nth Events X+D} >: 0)
             }}
             1}
         end
      end
   end
end

   proc {EventEvery List NumEvents}
      {ForAll {LUtils.everyNth List NumEvents}
       proc {$ X}
          X \=: 0
       end}
   end

   proc {NoRests List}
      {ForAll List
       proc {$ X}
          X\=:2
       end}
   end
```

```
proc {NoTwoIdenticalAdjacentBeats Durs BeatDivisions}
   {Pattern.forNeighbours
    {LUtils.everyNth Durs BeatDivisions}
    2
    proc {$ X}
       ( {Nth X 1} \=: {Nth X 2} )
    end}
end

proc {MinDursOnBeats Durs BeatDivisions DurValue NumDurs}
   {FD.sum {Map
            {LUtils.everyNth Durs BeatDivisions}
            fun {$ X} (X=:DurValue) end}
    '>=:' NumDurs}
end

proc {MinDurs Durs DurValue NumDurs}
   {FD.sum {Map Durs fun {$ X} (X=:DurValue) end} '>=:' NumDurs}
end
```

Functions retained in the individual level:

```
%% actual level definition
proc {GetEvents Sol}
   Durs Events
in
   Sol = Events#Durs
   %% setup defines the Event and Durs list
   {EventDurs.setup Beats BeatDivisions Events Durs}
   %% actual level definition
   {RhythmFuncs.eventEvery Events BeatDivisions}
   {RhythmFuncs.noRests Events}
   {SixteenthsTogether Durs}

   if (Interesting) then
      {RhythmFuncs.noTwoIdenticalAdjacentBeats Durs BeatDivisions}
      {NumDurs Durs}
      {HasEighthSixteenth Durs}
```

```
      {NoEighthEighth Durs}
   end

   {FD.distribute ff {Append Events Durs}}
end

proc {SixteenthsTogether Durs}
   {ForAll
    {Pattern.adjoinedSublists Durs 2}
    proc {$ X}
       {FD.equi
        ( {Nth X 1} =: 1 )
        ( {Nth X 2} =: 1 )
        1}
    end}
end

proc {HasEighthSixteenth Durs}
   {FD.sum
    {Map
     {Pattern.adjoinedSublists Durs BeatDivisions}
     fun {$ X}
        {FD.disj
         {FD.conj
          {FD.conj
           ( {Nth X 1} =: 1 )
           ( {Nth X 2} =: 1 )
          }
          ({Nth X 3} =: 2)
         }
         {FD.conj
          {FD.conj
           ( {Nth X 3} =: 1 )
           ( {Nth X 4} =: 1 )
          }
          ({Nth X 1} =: 2)
         }
        }
```

```
     end}
    '>=:' 1}
end

proc {NoEighthEighth Durs}
    {ForAll
     {Pattern.adjoinedSublists Durs BeatDivisions}
     proc {$ X}
        {FD.conj
          ( {Nth X 1} =: 2 )
          ( {Nth X 3} =: 2 )
     0}
     end}
end

proc {NumDurs Durs}
   {RhythmFuncs.minDurs Durs 1 6}
   {RhythmFuncs.minDurs Durs 2 2}
   {RhythmFuncs.minDursOnBeats Durs BeatDivisions 4 1}
end

Beats=4
BeatDivisions=4
BaseFilename='rhythms/sixteen02/'
```

B.3.2 Importing Exercises into MEAWS

Strasheela exported the exercises as LilyPond input files, which were then compiled into png (image) and midi files. The midi files were further processed to extract the pitches and times; the png files were left untouched. When MEAWS loaded an exercise, it displayed the png and set the expected answers to the onsets or pitches loaded from the file. The "exercise answer" files were simply text files. As such, it is very easy to create new exercises.

New exercises may be created by writing the sheet music in GNU/LilyPond. The `data/` directory of MEAWS includes a `header.ly` file to assist in creating a single system of music. Once the lilypond input file has been written, run the python script `makeexample.py` on the file. This will produce a `.png` and `.exp` file which may be loaded by MEAWS.

New games may also be created. The directory `data/game/` contains two sam-

ple games, `rhythms.txt` and `intonation.txt`. A new game may be created by using these files as the basis. Alternatively, a python script may be created to produce the game files. The same directory contains two python scripts, `makerhythms.py` and `makeintonation.py`. These scripts are easy to understand and may be modified to produce different games.

B.3.3 Discussion of Implementation

The disk space for each exercise is almost entirely spent in the png images. The images range from 1 Kb to 3 Kb, while the answer files are less than 0.1 Kb. There are approximately 350 rhythm exercises and 900 intonation exercises, producing a total data size of slightly less than 4 Mb. This is an appropriate size for such a program, but clearly a future version of MEAWS cannot contain a library of thousands of exercises if it is to remain within sensible download size limits.

One solution would be to render sheet music directly in MEAWS. Given my experience with GNU/LilyPond, I am reluctant to pursue this – producing high-quality sheet music is much more difficult than people realize. That said, technical exercises do not need to be beautifully engraved. If the data files only specified the notes to display, the size of the data could be reduced by almost 400 times.

Another possibility would be to download new exercises in the background. Assuming that MEAWS would only be used on a computer with a working internet connection – an assumption that is increasingly easy to grant – MEAWS could silently download a new set of 100 exercises every time the user was almost finished with the current exercises.

Appendix C

User Study

A user study was performed to gather feedback about MEAWS. Potential users were invited to download and try MEAWS on their own, and provide comments via an anonymous web form.

C.1 Qualitative Comments

In this study, qualitative comments ("The rhythm practice was good; I really improved" or "I couldn't figure out how to use the intonation exercises; I always got a red screen") were sought rather than quantitative data. The latter could have been provided if MEAWS sent data about user usage over the internet – this would have provided exact data about which exercises were attempted and what the scores were. Over time, this would provide solid evidence of the improvement (or not!) of users. However, tracking usage in this way would involve a greater invasion of privacy, so it was decided to perform a simple study first to establish whether future studies (with less privacy) are warranted.

C.2 Study Results and Comments

There were a surprising number of technical and user interface problems. As a result, only four people filled out the anonymous web form after successfully running MEAWS. Seven people sent email directly to me, discussing technical problems or expressing disappointment that the MEAWS intonation was only for violins and not for vocal training. Due to the design of the study, we do not know how many people downloaded MEAWS and ran it successfully without providing feedback.

Many people expressed disappointment that MEAWS was not available on Windows; due to my inexperience with Windows software development, MEAWS was only available on MacOS X and Linux. However, some users still have problems with the Linux version: it used ALSA (Advanced Linux Sound Architecture). One

user reported that (s)he could not record sound with MEAWS on his system running JACK (a low-latency audio server for Unix systems; according to this user it is "the way of the future"). We were surprised that the newer JACK system could not support ALSA applications (possibly with some emulation layer), but we have little experience with Linux audio systems so could not suggest any alternatives, nor provide a fix for MEAWS. A few users reported difficulty setting K_t, the threshold for clap detection.

C.2.1 User 1

This user is a skilled musician whose principal instrument is the clarinet, and has been playing and practicing music for over 20 years.

Rhythm Comment:[1]

> I had great fun trying the rhythm exercises – it was definitely addictive. I can't actually remember whether I reached "hard" or "challenging" as it's a while since I last tried them. I was surprised at the scores at first, but realized that the ability to be exactly on the beat is just as important as knowing how often to clap – these exercises helped me focus on keeping a strict beat.
>
> Although as a "skilled" performer I found the early levels easy (apart from the "staying on the beat" issue), I appreciated the way the exercises gradually led students to master increasing levels of difficulty.

Intonation Comment:

> I can't remember what level I got to – I think it was "simple", but it may have been "easy".
>
> Like the rhythm exercises, these intonation exercises were fun and very addictive. As a novice my results were somewhat random – having achieved a score of 89% at one level quite soon after reaching that level, I decided that this must have been luck, and I was correct – it took me many more attempts to get back to that level. I can't remember whether I would have been allowed to "move up" after that lucky try, but I don't think I should have been – I think students should have to get over 80% at least twice before going on to the next level.
>
> With these exercises to guide me, I'm even tempted to try to learn the violin again – about twenty years after I gave it up the first time.

[1] These comments are reproduced verbatim, including any spelling mistakes or typos.

A feature I would like to see included would be to hear the correct passage played by the computer before I tried it. On one occasion (the minor key exercise), I practised the exercise with F#s instead of Fs, so always got the error screen – if I'd heard the exercise first (or been able to listen to it after my first attempt), I would have realized my mistake sooner.

This will be a very useful tool for motivated beginners – I wish there was something like this I could use for clarinet or voice exercises. I think it will also appeal to those who are not normally highly motivated – the desire to beat one's previous score is strong in most people.

Response:

This user's confusion about the "minor key exercise" is understandable – the first four levels share the same key signature, and there is no explicit warning that the key signature has changed. Providing such a warning (either as a pop-up box, or highlighting the key signature on levels where it has changed) would be the easiest solution. Playing the correct audio for each exercise would be a fairly substantial new feature – either adding MIDI synthesis, recording hundreds of exercises, or adding pitch shifting to MEAWS as described in [22].

C.2.2 User 2

This user is a professional musician whose principal instrument is classical guitar, and has been playing music for over 20 years and practicing for between 10-19 years.

Rhythm Comment:

I liked the rhythm part very much. You really have to be accurate to line it up just right. The rhythm part is really nice, though. I think I'd like it better if I didn't have to go into a menu and choose each excercise manually, though. If it would automatically go to the next one that would be great. Maybe the user could choose a difficulty level and that would be the starting place, then each exercise after that would be chose by the software based on how well the user did. If s/he did very poorly, it would choose and easier one, and if s/he did well, it would proceed to harder ones.

Intonation Comment:

I don't think I got the intonation part to work right. I didn't see any instructions on it and I wasn't sure whether to try to do the intervals using relative pitch or exact A-440 or what.

Response:

This user apparently choose "Open Exercise" instead of "Open Game". The former option was added mainly for testing purposes; instead of beginning a game, it only opens a single exercise. This is an embarrassing user interface mistake on my part: most users will never want this function, so it should definitely not be part of the main interface. At the very least it should be hidden under an "Advanced" or "Testing" menu, if not disabled entirely when creating the finished version for public use.

Given the user's principal instrument of guitar and the question about "relative or exact A-440 or what", we suspect that the user was not attempting to play violin for the intonation portion. This is understandable: musicians know that intonation affects all instruments[2], so the division of "violin intonation" vs. "other instrument intonation" is not readily apparent. The user interface of MEAWS must be modified to take this into account – the audio analysis will not work on classical guitars, and the entire layout of the exercises (the gradual progression of easier to harder exercises) will not make sense for any other instrument.

C.2.3 User 3

This user is a skilled musician whose principal instrument is the piano, and has been playing music for between 10-19 years and practicing for 1-3 years.

Rhythm Comment:

> I've spent several mintues trying to pass the first level, just to find the good parameters. Then I passed all levels.
>
> - (important) you should make it clear that the intro is 4 ticks (and that the 1 tick is when the user hits "space", and not the following tick). Perhaps a visual indication "1/2/3/4" on the metronome could help (just by looking on the blue/red balls, I've no idea where is the strong beat). (The "align first beat" option seems intersting, but it seems to fail with syncopas : it aligns the first clap on the first beat, and not on the first clap of the exercice).
> - (important) the rhythm should end on the 1st quarter of the following mesure, epecially for easy levels : for a beginner, I think it's very difficult to clap
>
> c4 c4 c4 c8 c

[2]Even pianos can be out of tune, although fixing intonation on a piano may require relatively unusual skills and tools.

finishing on two eights. The rythm
c4 c4 c4 c8 c | c4
seems me more fair.

- (suggestion) you could add more difficult levels. Perhaps 2-voice rythms, with simultaneous tuplets + eights ? Of course, in this case you could check only the global rhythm : only the exercice display to the user will be "polyphonic".

Intonation Comment:

I'm sure it will be someday a great tool for human voices :)

Response:

This user is mistaken when (s)he writes "the intro is 4 ticks"; MEAWS shifts the detected claps as appropriate. The introduction could be 40 ticks if the student wished it to be so. It understandable that a musician may think an introduction was 4 or 8 beats, since the entire exercise was in 4/4 time. Still, evidently some combination of the user's background knowledge / expectation and the MEAWS interface suggested to him or her that this "4 ticks" was an absolute requirement. The MEAWS user interface should be modified to correct this misunderstanding.

The user's report about "align first beat" not working if an exercise starts with a rest is quite valid, however. This bug must be fixed before further testing is attempted.

The user's suggestion to add an extra note – the downbeat of the next bar – is an excellent idea. It parallels the "courtesy 1" which we added to the Event and Duration lists in Chapter 2.

The user's final suggestion, that we add more rhythm levels, is certainly a good long-term idea. However, in the short term I will focus on fixing the other problems and beginning more widespread testing of MEAWS.

C.2.4 User 4

This user is a skilled musician whose principal instrument is the violin, and who has been playing and practicing music for between 10-19 years.

Rhythm Comment:

I found it a little hard to stay with the metronome. Some sort of analogue (or subdivided) metronome would have been easier to follow. Also, I thought rushing (or slowing down) was heavily penalised compared with simple wrong rhythms.

Rhythm Comment:

I dislike equal-tempered tuning very much, particularly when playing open fifths! Next time, I'll change the settings to use "Just intonation."

I had trouble getting a low B (below middle C) recognized; I had to play that exercise several times since I kept getting a yellow background.

Response:

As described in Chapter 3, we insist on following the metronome's tempo in order to simplify the analysis. In real music, following an external tempo may or may not be so important – when playing solo or with a small group, the tempo may vary. However, when playing in an orchestra with eighty people, one person rushing or slowing down causes more problems than simply omitting some notes in order to stay at the group's tempo.

We are not certain what prompted the user's difficulty with the "low B" – the pitch analysis should have no difficulty with that note. It is possible that the user was playing level 7, which requires a B♭ instead of B♮. The only information about this ♭ is given in the key signature, but MEAWS gives no explicit warning about the key signature changes, as we discussed with User 1.

C.3 Discussion of Study

Due to all the technical problems and misunderstandings about the user interface, this user study was quite valuable. Programmers are notoriously bad at guessing how other people will attempt to use their software, and we were no exception.

Based on the technical difficulties with MEAWS, we are seriously considering abandoning the C++ implementation and rewriting it in Flash. Flash 9 supports microphone input in addition to simple graphical user interfaces, so the rhythmic exercises could be performed in this language. Instead of downloading MEAWS, users would simply visit a web page, wait a few seconds for the program to load, and then begin clapping.

Unfortunately, this would split MEAWS into MEAWS Rhythms and MEAWS Intonation. Calculating the RMS values required to grade rhythmic exercises should pose no problem, but Flash does not contain any built-in pitch detection, and we doubt that the language is fast enough to perform Fast Fourier Transforms with any degree of speed. However, such a split may not necessarily be a bad thing. The violin intonation exercises are only useful for beginning violinists; singers who attempt to practice their sight-singing using MEAWS intonation (as it currently stands) will be sorely disappointed. By splitting MEAWS into rhythmic exercises implemented in

Flash and violin intonation exercises implemented in C++, we can hopefully reduce the number of people mistakenly attempting the intonation exercises.

The interest in sight-singing exercises was surprising – half of the survey participants expressed an interest in it, as did over half of the people who emailed me directly. In retrospect, this should not have been startling: there are many, many more amateur singers than beginning violin students. In addition, there are many more "complete beginner" amateur singers than there are "complete beginner" violinists – buying a violin costs hundreds of dollars, and thus discourages extremely casual would-be musicians from beginning to play violin.

To accommodate this interest, the audio analysis of MEAWS would need to be improved somewhat. Singers – especially amateur singers – do not produce as constant a pitch as violinists do, and note boundaries are not as abrupt as we see on a violin. The note segmentation would need to be a bit more sophisticated (potentially even marking 3-10 frames as "boundary material," not belonging to any note), but this is certainly within the current capabilities of music transcription. In addition, an entirely new set of exercises levels would need to be created for singers. The music representation would be much easier than the violin intonation, however: a list of Pitches and possibly a list of Intervals is all that would be required.

Finally, some amount of automatic configuration for K_t should be used. Perhaps the first time the user runs MEAWS, it should ask the user to clap four times. MEAWS would then analyze the audio and pick the largest four peaks, and compare their values to the remaining peaks. If it can find a threshold value which makes sense (say, the top four peaks were all above 0.25, while the remaining peaks were all below 0.15), then MEAWS should automatically set $K_t = 0.2$.

In addition, if the user appears to have clapped the wrong number of times, MEAWS should attempt to re-calibrate K_t. For example, if the expected number of claps for a particular exercise was 12 but the user only appears to have clapped 11 times, MEAWS should automatically attempt $K_t = 0.15$ and $K_t = 0.10$. If one of those values produces 12 detected claps, then MEAWS should accept this new value of K_t and grade the exercise accordingly.

C.4 Conclusion

The user study successfully identified a number of problems that must be addressed before undertaking a large quantitative study. In addition to various user interface tweaks, we shall seriously consider splitting MEAWS into separate programs for rhythmic and intonation exercises, as well as adding intonation exercises for singers.