Improving Interaction in a Musical Tutor for Playing by Ear

Aleksandrs Zavalnijs

Master of Science

Computer Science

School of Informatics

University of Edinburgh

2011

Abstract

There are many people who at some point of time decide to learn music by themselves, skipping years of musical school. While learning sight-reading is one of the musical skills which can be developed quite easily, there are some which can not. One of such skills is playing by ear. The reason why it can be so tough is that one needs some kind of curator to notice any mistakes and provide efficient feedback. Of course, one might want use software tutors; however, most of them provide immediate right-or-wrong feedback, which can be very limiting. Because in such interaction student is interrupted once his first mistake is made, student cannot use his own hearing and decide whether he is right or wrong by himself. This work is about developing an improved and more efficient interaction in musical tutor named "EarCog" which gives more freedom to a student and allows him to use self-judgment. In the end, hypothesis that such interaction is indeed more efficient is being tested by conducting an experiment.

Acknowledgements

Many thanks to my helpful supervisor Alan Smaill.

Thanks to Barbara Webb and Karen Ludke for advices on testing process setup.

I am also very grateful to my old friend and musical teacher Stanislav Sheiko for musical consultations as well as for providing test subjects.

And, of course, I would like to thank all test participants for their time and effort.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Aleksandrs Zavalnijs*)

Musical glossary

The following definitions provided from "Virginia Tech Multimedia Music Dictionary" [Philpott, 1996].

Aftertouch — ability in musical keyboard to change dynamically tone of sound by depressing the key; also called "keyboard expression"

Broken chord — a chord in which the notes are not played simultaneously but rather they are played successively.

Chord — the sounding of two or more notes (usually at least three) simultaneously.

Harmony — the combination of notes sounded simultaneously to produce chords; countermelodic notes to accompany a tune.

Interval — the distance between two pitches.

Melody — a tune; a succession of tones comprised of mode, rhythm, and pitches so arranged as to achieve musical shape, being perceived as a unity by the mind.

Note — a notational symbol used to represent the duration of a sound and, when placed on a music staff, to also indicate the pitch of the sound.

Octave — an interval spanning seven diatonic degrees, eleven semitones. An octave above C would be C. The frequency of a note one octave above another will have exactly twice as many Hertz as the frequency of the note an octave below it.

Pitch — The specific quality of a sound that makes it a recognizable tone. Pitch defines the location of a tone in relation to others, thus giving it a sense of being high or low.

Scale — a series of notes in ascending or descending order that presents the pitches of a key or mode, beginning and ending on the tonic of that key or mode.

Score — the entirety of the instrumental and vocal parts of a composition in written form, placed together on a page in staves placed one below the other.

Tempo — the speed of the rhythm of a composition. Tempo is measured according to beats per minute.

Velocity — volume of played note or chords; depends on how hard key is pressed on piano or synthesizer; not all synthesizers support velocity

Table of Contents

Chapter 1

Introduction

There are many people who decide to learn music at home, without taking private lessons, or attending to musical school. While learning the sheet music, i.e. playing from musical score, is already not an easy task, playing by ear is considered among public as even harder skill to develop.

This skill is usually being developed in musical school during long years of learning each musical interval and is tested by the exercise called "sol-fa" in only last years [Chimbombi, 2007]. Therefore, it is quite a challenge for people who are learning on their own — it is hard to notice mistakes, especially at the very beginning of learning process.

Due to the absence of any available assistance at home one might consider using a computer program for self-teaching to play by ear. In short – the process is simple: the user has some MIDI device and computer and uses the program which receives information from MIDI device and gives some feedback. And here exactly we face the problem – while there are lot of programs that tackle this task, the functionality provided by them is very limited. To be more specific, the main issue with them is in the way feedback is organized. Most of the time, it is just comparing the correct melody with melody played by student note by note, and interrupting him on the very first mistake made.

There are a few clear reasons for such implementation: such way of interaction with student seems very obvious and implementing it is a very easy programming task. However, such tutoring, no matter how well all the exercises are prepared, does not seem to be very helpful simply because it does not allow student to rely on his own hearing.

This work proposes new, more comfortable and loyal feedback, which is called here "deferred judgment feedback". The main principle of it is allowing mistakes, like wrong notes, repetitions, extra notes or skipped notes, by carefully following where exactly student is currently playing and giving him a chance to figure it out and showing his performance detailed results only once requested. The aim of this work was to test such way of interaction against the common one to find out whether is has any advantages.

After testing and comparing both ways of interaction on two randomly divided groups of students results were positive and conclusions were that proposed interaction shown itself as more helpful way to learn playing by ear. Of course, research conducted was too small to make big claims and more extensive testing is required (see chapter 6).

The "Background" chapter of the dissertation contains all the theory: the research of how exactly this problem is tackled at musical schools, examination of advantages and disadvantages of all existing solutions, the description of how exactly proposed feedback is designed as well as research on algorithms suitable for this task and choosing most appropriate.

In the "Elaboration" chapter the practical side of thesis is covered: it goes from top-view software design down to the implementation of each part with description of problems faced and solutions taken.

Then, the "Evaluation and analysis" chapter contains all information about testing developed program on two groups of students, collecting and comparing their results and performing statistical analysis to make sure that data is statistically significant. Also, students answers to questionnaire are discussed.

All results summary and conclusions made from them are presented in "Results and conclusions" chapter.

Finally, "Future work" chapter describes how this work could be extended and what could be done to make testing more extensive and functionality of designed software richer.

Chapter 2

Background

This chapter contains all the theoretical preparation for developing software musical tutor. That includes examination of how playing by ear is tough at musical schools, research on existing software solutions and formulating the hypothesis. After that, the proposed feedback is discussed in details as well as algorithms suitable for implementations are identified. The practical side of this work is described in next chapter, "Elaboration".

2.1 Examining musical schools

Before analyzing software possibilities for problem tackling, the real world analogy was analyzed – the musical schools themselves. To do this, a few musical teachers as well as musicians who finished musical school were asked to describe how musical schools are helping to develop this skill.

First of all, it turned out that this skill is "tutored" in very last years, after all musical theory is studied. The word "tutored" was put in quotes, because there is no real tutoring process, only testing. The reason for that is that by that time, after long years of learning each interval, this skill is already developed. It is tested through sol-fa dictations — students listen to a melody and they need to write it down [Chimbombi, 2007].

Second, there are two types of "musical ear", that are distinguished in music — relative and absolute ones. The relative one is when you can hear the interval (i.e. distance between pitches), the absolute one is when you distinguish the pitch itself. The last one is considered much harder to develop, so in this work the concentration is only on first

one, relative musical ear. This is done by always giving first note of sequence and asking to figure out the other ones. Last thing to stress is that these dictations are then checked by teacher, marked and given back, which makes this interaction pretty slow.

So, to sum up, the skill of playing by ear is not tutored at musical school directly and specially. To get to the part when you start writing dictations, you need to learn all the theory and remember all intervals: musical schools are working in a very traditional approach. However, it is known that there are professional musicians out there who can play by ear, but do not know the notes, so this knowledge, although is definitely helpful, does not seem to be necessary.

In any way, one might just want to learn playing by ear at home and, because it is hard to do on your own, replace real musical teacher with a software one. The research on existing solutions in this field is presented in section 2.2.

2.2 Research on existing solutions

At first, we need to decide what software we need to analyze. Then, this software is analyzed in the table with advantages and disadvantages enlisted. In the end of this section some common problems of software found are summed up.

2.2.1 Formulating requirements

Actually, more or less all types of software which fall into the category "playing by ear tutoring" are examined. To be more specific, such software needs:

1. Functionality of playing melodies
2. Receiving sound from some input (MIDI device, mouse, microphone)
3. Checking and scoring an attempt
4. Some kind of increasing difficulty of exercises.

2.2.2 Examining current products

There are many solutions that are trying to attack our problem from different angles. Most products are either online solutions or installable programs. Obviously, it is impossible to analyze and examine all of them. In the table below you can find some examples of such programs which author was able to find on the web.

Table 2.1: Existing product analysis

Product name	**Advantages**	**Disadvantages**
Play by Ear for iPhone[Mayers, 2011]	• possibility to use piano (pitch extraction)	• interrupting right/wrong feedback
TrainEar[Ruska, 2011]	• no software needed to install	• interrupting right/wrong feedback • only playing with mouse is supported
Good ear[Schoeberl, 2011]	• no software needed to install	• interrupting right/wrong feedback • only playing with mouse is supported

Product name	Advantages	Disadvantages
HappyNote[Riben, 2011]	• game-like mode	• interrupting right/wrong feedback • hard installation process • only playing with mouse is supported
Ear Training Coach[Adventus, 2011]	• different levels of difficulty • game-like interaction • very good lessons organization	• interrupting right/wrong feedback
Ars Nova Practica Musica[ActiveMusician, 2011]	• large musical exercise database • Chords and intervals ear training	• interrupting right/wrong feedback
EarMaster Pro 5.0 Software for Aural Test Practice For PC [Havvej, 2011]	• Rich progress reports • Microphone notes recognition • Support of singing, humming, clapping.	• impossible to train on melodies, only on intervals

2.2.3 Common problems

Although there are lots of rich and interactive products for playing by ear skill development out there, there is one common problem with them.

While all of them provide a way for student to check whether he plays right or wrong, the

way it is implemented is very interruptive. Student sees some sort of message after first mistake without getting a chance to work it out by himself. In this dissertation we are calling it "instant judgment feedback". In the table 2.1 "Existing product comparison", some of such programs examples are shown. The last one, which is one of the most advanced software products, also uses such strategy (see figure 2.1).

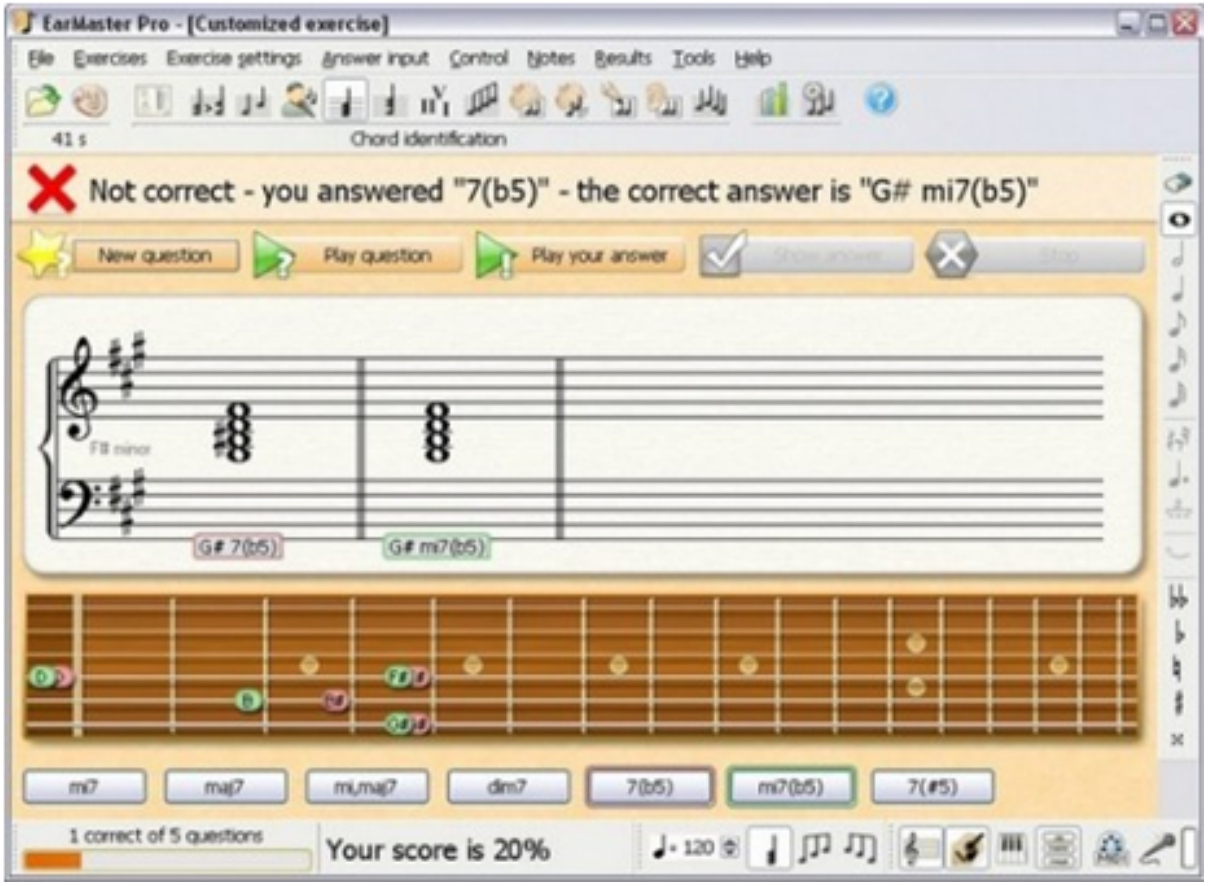

Figure 2.1: EarMaster Pro screenshot. Right-or-Wrong instant feedback.

The "instant judgments feedback" gives certain results and trying to disprove it is by no means a purpose of the dissertation. Probably it has been used in most, if not all, ear training suites for this reason: it is easy and obvious to implement and it works. The main question that this dissertation tries to answer is – are there any ways to improve this interaction method to increase the efficiency of learning to play by ear?

2.3 Defining hypothesis

Author of this work has an experience of using "PlayByEar" application on the iPhone. This software teaches to play by ear and also uses same interruptive feedback. The only comfortable way to figure the melody out, which author was using, is closing the microphone hole with the finger, while making few attempts, and then releasing the microphone to play for marking, which was obviously against designed way of usage.

With such interaction student cannot use his own hearing to notice mistakes, because he is notified by software once mistake is done. However, the aim of learning feedback should be to train a student in self-feedback and in self-judgment[Draper, 2005], e.g. to be able to identify mistakes using hearing. To make it possible, immediate judgment should be postponed to give student a possibility to figure out melody on his own by listening to his results. In this work, such feedback is named "deferred judgment feedback", and the dissertation hypothesis is formulated below.

Hypothesis: It is possible to achieve more efficient learning of playing by ear using software musical tutor by implementing "deferred judgment feedback" instead of "instant judgment feedback".

Aim: To design and develop a tutoring program with both interactions implemented and conduct an experiment to check the hypothesis.

2.4 Proposing deferred feedback

Design and specification of proposed feedback:

Commonly used feedback: comparing every melody note with input; the attempt is marked as "correct", if all the pressed keys were right, or marked as "wrong" once the first mistake is done.

Problem: Such feedback is limiting musician in that sense, that it does not allow him to use his hearing for mistake detection, and cuts him off during this rather creative process.

Main requirement: the "deferred feedback" should allow musician to figure out the melody and not stop them, if he presses a wrong key.

Implementation details: using any suitable algorithm (see section 2.5) to detect the place where most likely musician is currently playing (i.e. where he thinks he is playing), and make also comparisons with right notes, which would allow to provide detailed analysis. Do not interrupt on wrong notes, instead show his results on request ("SPACE" key press on keyboard)

2.5 Potential suitable algorithms comparison

So, for following where musician is currently playing, some algorithm was needed. The requirement was that the algorithm should "take" two sequences and say, where the tail of second sequence is situated in the first one. After some research, three candidates where found: Levenshtein distance, sequence alignment and Hidden Markov models .

Levenshtein distance is a metric for measuring the difference between two strings of text [Bard, 2007]. In its common implementation, a matrix is created for comparing each letter from one string with each letter from another string and adding or subtracting some points from previous matrix row to get the resulting score of how much two strings differ. While it may sound like not suitable for this work's purposes, with some tricks it is possible to use this algorithm with fuzzy string searching [Miller et al., 2009]. Therefore, by replacing characters with notes and strings with melodies, it is theoretically possible to use it for finding current place of playing — it deals very well with substituted, inserted and removed characters — in our case it would be wrong, extra and missed notes. However, it is not designed to deal with repeating subsequences — in our case, the melody can be half-played lots of times before getting to the end.

Sequence alignment algorithm is mostly used for biological purposes to align sequences of DNA, RNA or proteins and find any similarities between them [Kim and Kececioglu, 2008]. It could be possible to use this algorithm for score following, for example, with Dynamic time warping algorithm, but again, it is used to align two sequences that vary in time, but, as already mentioned, cases of multiple replaying same melody part can occur [Chu and Li, 2011]. It was dismissed also because it does not provide any information about the probabilities of sequence matching.

The algorithm on which the research was stopped was Hidden Markov models , due to its

absolute matching this work's needs: it provides probability calculations as well as representation of sequence of states of however characterized [Orio et al., 2003] [Pardo, 2005]. More about Hidden Markov models in section 2.6.

2.6 Background to MIDI

MIDI, or Musical Instrument Digital Interface, is a protocol for electronic musical instruments communication [White, 2000], defined in 1982. Essentially, instead of sending analog signals, it is based on event information exchange. Among many event types, like `Poly Key Pressure`, `Controller Change`, `Program Change`, `Channel Pressure` and `Pitch Bend` most popular are `Note on` and `Note off` events, signaling that some note was pressed or released. In `Note on` event channel, frequency and velocity (volume – how hard the key was pressed) are passed. There is also rather specific event called `Aftertouch` which happens when musician changes the pressure on key while holding it. Such technique is considered to be an advanced one and will not be used in this work.

Same concept is also utilized in MIDI files with additional serialization features. A MIDI file's main musical structure is a track and each track consist of chunks of MIDI events. In current work, only self-prepared MIDI files were supposed to be used, so only first track of MIDI file is read by default. As mentioned earlier, there are many types of MIDI event, but in this dissertation three of them were meant to be used — `Note` on, `Note off`, described earlier, and `Tempo Change` event, which notifies that the tempo of track was changed. Tempo is, basically, used to convert ticks (MIDI virtual time) to milliseconds.

2.7 Research on Hidden Markov models

Hidden Markov model (HMM) is sort of state machine and given possible states and observed sequences it can be used to determine most likely sequence of hidden states that caused an output of observed ones [Stamp, 2004].

The most common applications of Hidden Markov models are recognition of speech [Gales and Young, 2007], handwriting, movement, partial discharges, cryptoanalysis and machine translation. A slightly less usual way of applying Hidden Markov Model is musical score following. For this work, Hidden Markov Model is used similarly to the way

it was used in work "Artificially intelligent accompaniment using Hidden Markov Models to model musical structure" [Jordanous and Smaill, 2008]. In their work, authors used Viterbi algorithm in Hidden Markov Model context to determine what place of musical piece musician is now playing.

To understand Viterbi, it is better to figure out, what an HMM essentially is. As César de Souza described in his article,

> "Hidden Markov Models can be seem as finite state machines where for each sequence unit observation there is a state transition and, for each state, there is a output symbol emission" [Souza, 2010].

And, basically, having possible hidden state sequences, probabilities of transition from one state to another, we can take any *observed* sequence and calculate, what is the most likely sequence of hidden states that generated this sequence. And this is exactly what Viterbi algorithm does [Forney, 1973]. So taking musical melody as hidden state sequence, where state is each chord in this melody, and taking played by musician melody as an observed sequence, we can spot what place of melody is most likely currently being played.

On the topic of practical application of Hidden Markov models , please read section 3.4.

2.8 Choosing marking method and algorithm

Another question to answer was how to perform student's assessment method, which would be used for before and after learning phase examinations. After musical teacher's consultation the decision was made to use similar to real musical schools' methodology. In musical schools, students listen to the melody twice and then write down heard piece of play in musical notation. In our case, they would listen to the melody twice and then, when ready, play the melody for assessment. After that, amount of errors would be calculated.

At that point, the question about how to implement the scoring function arose. The most obvious solution to compare sequences of notes (in both correct melody and received from MIDI input) note by note turned out to be not sufficient. Although it would handle wrong notes, it wouldn't cope with missed or extra notes. For example, if the melody would be "C D E F" and played notes would be "C E F" it would counted 3 errors (all notes after "C" are not in their places). However, one of the algorithms reviewed for the role of score following algorithm turned out to be suitable: Levenshtein distance.

As already described (see section 2.5), Levenshtein distance perfectly deals with all kinds of edits. Not only it handles missed, wrong and extra notes, but it also gives count of differences as a result of calculation, and it was exactly, what was needed.

2.9 Chapter summary

In this chapter, all theoretical part of the work was presented. To examine playing by ear skill, at first, some research on musical schools was conducted. However, as it turned out, this skill is not developed directly there, and is improved more like a side effect through long years of learning musical theory — sol-fa dictation testing this skill appears only in last musical school years.

For one who prefers learning at home, the only solution to speed up the process of learning to play by ear is to use some soft of software musical tutor. A special research on existing solutions was done, and some specific common problem was found — student cannot use his hearing because he is interrupted and assessed once mistake is done, even if student himself knows that he has pressed wrong key. Instead of immediate judgment feedback, deferred one was proposed for implementing and then testing.

After the feedback was designed in details, and exact requirements from the software became clear, survey on suitable algorithms was conducted and Hidden Markov models were chosen as the most appropriate. In the end, Hidden Markov models and MIDI format were studied in depth.

Next chapter, “Elaboration”, is covering the practical part of this dissertation.

Chapter 3

Elaboration

This chapter is a description of practical part of the thesis. That includes top-level design description of the "EarCog" system, explaining choice of development tools, specification of how all the algorithms are used and how modules are implemented for this work.

3.1 Top-level design

Before diving into how each part of the system is built, at first, let's see how the system *should* work and then, from which parts it should consist. So, the global view of system's functionality is presented in the following list:

1. Input from synthesizer or MIDI keyboard is read;
2. There is a possibility to choose input and output devices to work with;
3. Exercises are read from MIDI files;
4. Feedback type can be switched; to test the given hypothesis, we should test the efficiency of both "instant judgment feedback" and "deferred judgment feedback", proposed in section 2.4;
5. The possibility to output the sound is presented (most MIDI-keyboards do not have speakers);
6. In exam modes, students attempts are marked and their results are logged to file;

7. Once setup is done, system is receiving MIDI input from devices and processing it; the algorithm of processing depends on feedback type selected;
8. If "instant judgment feedback" is selected, on the first mistake or once correct melody is played results of attempt are displayed.
9. If "deferred judgment feedback" is in use, system detects, where student plays now, and shows results only once requested (by pressing "SPACE") or once correct melody is played.

To implement all these features, following modules of "EarCog" program were identified:

1. module for working with MIDI files, recognizing input from MIDI device, playing MIDI to output device
2. module with feedbacks implementations
3. module for writing log file
4. module for marking student's exams

3.2 Development tools choice

As development language, `C#` was chosen due to author's programming experience as well as due to wide availability of libraries for HMM and MIDI (see section 3.4 and 3.3), therefore as development tool `Visual Studio C# 2010 Express` was used. Microsoft `.NET Framework` version 3.5 was chosen as a framework.

3.3 Working with MIDI

For developing this dissertation product, both working with MIDI input/output and MIDI file were required. For MIDI input/output, MIDI.NET framework developed by Tom Lokovic was used [Lokovic, 2009]. This library provides classes for retrieving MIDI input/output device list, connecting and exchanging messages with these devices, as well as some business logic classes for notes, chords and scales with scale recognition functionality. After connecting to specific MIDI input device, further message retrieval process is organized via callback functions.

As a library for working with MIDI files and extracting information from them, NAudio component library was utilized [Heath, 2011]. NAudio is an open source .NET library for working with audio and MIDI. It covers all from playing music and receiving audio from input source to reading/writing audio files (like `.wav` or `.midi`), however, in this dissertation it was used only for reading, and not working with input, due to author's subjective opinion on imperfect NAudio library inner structure.

3.3.1 ASIO Driver For WDM Audio utilization

Once code for working with MIDI input was written latency problem was identified — there was a delay between pressing key and receiving this event in "EarCog".

After some research, it turned out, that the reason of such latency was in Microsoft's DirectSound drivers, which are not meant for musicians' purposes, and that for low-latency and high fidelity signal transfer, ASIO (Audio Stream Input/Output) drivers should be used. The main feature of ASIO driver is that it bypasses all operating system's layers of audio transfer and allows application to directly connect to the sound card. The problem with ASIO is that not all soundcards support it.

This issue was solved by installing ASIO2KS driver [Erichsen, 2002]. ASIO2KS is very sophisticated low-latency ASIO driver for all Windows Driver Model compliant devices. ASIO2KS, after installation, allows to choose, which audio devices to "convert" to ASIO emulated. Once its done, new virtual device is registered and can be used in applications and provided much lower latency. This driver was installed on all test machines along with "EarCog" program.

3.3.2 Threading issues and solutions

While implementing module, which performs all MIDI-related tasks, like getting notes from synthesizer and playing them to some audio output device, two problems related to threading where encountered.

3.3.2.1 Cross-thread control thread safety

The first problem was faced during development while changing the state of visual components of Windows Forms framework on MIDI input callback function call. MIDI-input callback function are called from second, non-UI, thread, which essentially is endless loop with interruptions on MIDI device. Such organization allows second loop to sleep, while UI of main application stays fully functional. Due to the thread-safety violation, while trying to access UI controls from non-UI thread, the "Cross-thread operation not valid" exception is raised.

After small research on topic, two workarounds for updating UI components from other threads were found. First workaround includes using `BackgroundWorker` class utilization and its `ProgressReport` event, which "happens" on UI thread side, therefore solves the problem. The second workaround is using delegates and `WindowsForms` control's `invoke` function, which allows to delegate some function call to UI-thread execution [Richter, 2006]. Second method was chosen due to absence of small, still possibly significant in future, class creation overhead in time and memory.

3.3.2.2 Delay in sound output after heavy processing

The second problem was related to delay in sound output and encountered after adding some processing (see section 3.4). The reason of delay was quite obvious. When MIDI-input thread receives some MIDI-event and calls the callback, it does it synchronously, thus, it waits for callback function return to continue MIDI input events retrieval. So, when callback function is still working, and new MIDI event happens, it is not being processed immediately.

To solve this issue, a third thread was introduced in the application. When MIDI-thread calls callback function with some MIDI event information, this event is put in event queue, sound is played immediately and callback returns, allowing MIDI thread to continue receiving input, while processing thread performs all event processing and calculations in background.

To make this solution even safer, this processing thread after creation is assigned to lower priority, so that processor would not give much of its time, even if calculation is rather long. As a result, there was no audible latency in playing sound anymore.

3.4 Using Hidden Markov models and Viterbi algorithm

To work with Hidden Markov models and Viterbi algorithm in C#, a ready library written by César de Souza and described in his web article "Hidden Markov Models in C#" was used [Souza, 2010]. The described code was recently incorporated in `HiddenMarkovModel` class of `Accord.net` library. In short, this class gives provides functions to work with Hidden Markov Models:

1. `Evaluate` — for calculation of likelihood that some observed sequence was generated by the model (where model can be seen as a set of hidden state sequences and transition probabilities matrix);
2. `Decode` — for finding the most probable hidden state that generated given observed state (Viterbi algorithm);
3. `Learn` — which takes an array of observed states and adopts out all other parameters, like probabilities of transitions between different states.

For current project only second function, `Decode`, is used.

As described in section 2.7, the Hidden Markov model is defined by array of possible states, symbol vocabulary, transition probabilities between these states, probabilities matrix of outputting each symbol by each state (emission probabilities) and an array of probabilities for each state to be a start state. The next subsections describe, how each of these parameters was configured in "EarCog" program, as well as explanations of made decisions.

3.4.1 Possible states

Each state in "EarCog"'s HMM is a note in melody (we were not dealing with chords) or ghost "mirror" of this note. Ghost states are introduced to deal with mistakes. The idea of ghost states was borrowed from the work "Score Following Using Spectral Analysis and Hidden Markov Models"[Orio and D'echelle, 2001] and slightly modified to fit current work needs. The idea is easier to grasp from the figure 3.1:

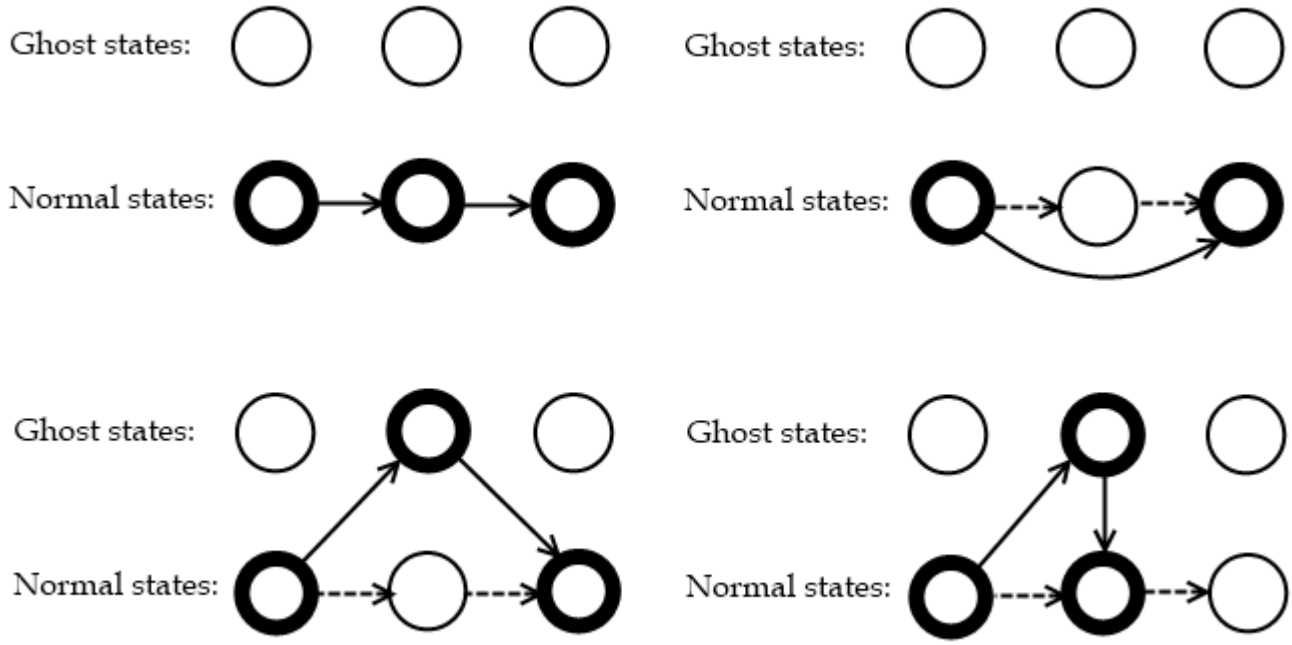

Figure 3.1: Ghost and normal states transitions. Taken from [Jordanous and Smaill, 2008] and restorated.

As shown on the diagram, ghost states handle wrong and extra notes, so that from every normal state there is a transition to next notes (for the reason why there are transitions not only to next note, but also to the notes after it, see subsection 3.4.5) as well as to the ghost notes that do not exist in the melody and will be visited in case of mistake in performance.

So, the amount of total states is N*2, where N is total number of notes in melody.

3.4.2 Symbol vocabulary

Symbol vocabulary is a set of possible symbol emissions (i.e. notes in current work) from all states. Because of the ghost states that can handle any wrong key presses, this set equals all possible notes can be received from the input.

3.4.3 Emission probabilities

Emission probabilities define how probable is for each state to emit each of the symbols in vocabulary. This part is not tricky – each normal state can only emit one symbol – the note which it corresponds to, and for all other symbols the probability is zero. Ghost state is a reverse of normal one – the probability of correct note is zero for it, and for all others are more then zero. After some experimenting, author came up with decision to make probabilities of wrong symbols gradually decrease in both sides, because it is less probable, for example, to make a mistake 10 notes away from correct one than 1 note away.

3.4.4 Initial state probabilities

This parameter is used in HMM calculations, when first observed state is analyzed, to decide, in our case, where musician was at the start. Although for professional score following it would possibly be enough to put probability "1" at the first note, and zeros on all others, for the purpose of current work, each place in the melody can be a start. This is done, at least, because after listening to a melody, student can, for some reason, remember only last few notes, and try to figure them out. In this case, first pressed note will not be first note of melody, therefore, every "state" should have higher than zero probability.

The probabilities, however, gradually become smaller with every melody note, to help HMM make right choices in case (which is not rare at all) of many repeated notes in melody: because its more natural to play from beginning, we are assuming that it is also more probable to expect.

3.4.5 Transition probabilities

Configuring the way transition probabilities are generated was the trickiest one. First, let's see how transitions themselves (without speaking about probabilities) are defined in current work. Author used ideas from "Score Following Using Spectral Analysis and Hidden Markov Models" paper [Orio and D'echelle, 2001] with some modifications.

In their work, to handle missed note, multiple transitions from ghost state were used. Unfortunately, there is a disadvantage of using such method for current work's purpose:

exact “position” detection. The problem is seen in following example: we have a melody “C D E F”, and student plays “C E F” sequence skipping the “D” note. Here is what would happen in case of using described method: 1) “C” state is visited, 2) ghost state of state “C” is visited (for “E” press) and 3) transition from ghost state to “F” note state is made (see figure 3.2). The reason why is happens, is that there is zero transitions from not sequential notes, therefore, ghost state must be in use. And, while this is not a problem for such purposes, as virtual accompaniment (because error rate is lower), it is an issue indeed for problem like “EarCog”, where not so many notes are in melody (for teaching) and to correctly detect each state is rather crucial.

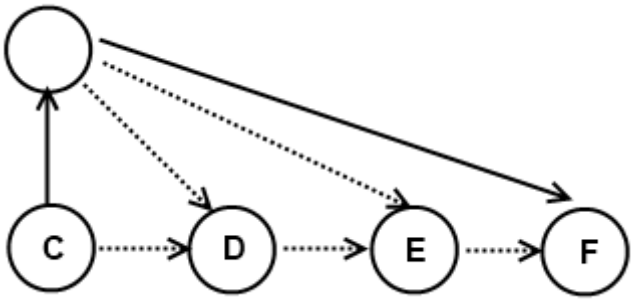

Figure 3.2: Melody “C D E F”. Playing “C E F”. Single transition between normal states

This is why not only multiple transitions from ghost state to following notes are defined, but also from each note there is a transition to all sequential notes. In this case, transitions through *states* “C”, “E” and “F” in described example are absolutely possible (see figure 3.3). It should be mentioned, that such decision also leads to certain disadvantage — the amount of HMM calculations required for observed state decoding. However, due to relatively small amount of notes in melodies supposed to be used (therefore, small amount of generated states), this is not a problem.

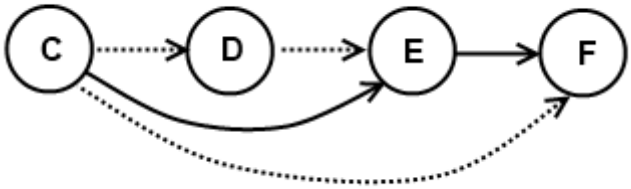

Figure 3.3: Melody “C D E F”. Playing “C E F”. Multiple transition between normal states.

Second, the probabilities of all the transitions were to be chosen. There are too many ways to arrange them, so, due to absence of much time for experimentation, after small testing first worked solutions were chosen.

From every note there is a transition to its ghost state as well to next note and all sequential notes. The highest probability has a transition to next note, and probability gradually lowers for all next notes. Something similar happens with ghost states: most probable is transition to next normal state (i.e. 1 extra note case), and transitions to all sequential notes are becoming lower. Ghost state has also a transition to itself, smaller than to next normal state — this is quite subjective decision, cause it is hard to say, whether, after one mistake, the next press is more likely to be wrong, than right. In any way, the aim of this work was not defining optimal parameters of HMM — instead, HMM was a tool to implement interaction for testing hypothesis.

It is worth mentioning that HMM can also be trained to adjust its transition probabilities based on observations, however, this functionality is not covered in current dissertation (see section 6.3).

3.5 Marker module

A special module for marking student's examinations was needed. The only requirement for this module was the ability to determine amount of errors made in particular test, given the correct note sequence and the one received from MIDI device.

At first, simple comparison (note per note) was planned to be used, however, more complicated algorithm turned out to be required for handling missed and extra notes, which is quire hard to implement through simple array comparison.

As already mentioned in section 2.8, "Levenshtein distance" algorithm turned out to be perfect solution for this task [Bard, 2007]. The pseudo-code of Levenshtein distance was borrowed from Wikipedia [Wikipedia, 2011] and translated to C# language. This code compares two arrays of chars and returns amount of differences between two sequences. For this work's purposes, sequences of chars were replaced with sequences of `Note` instances, which are basically strings.

3.6 Logging module

This module is the simplest module implemented, yet important for whole system working. During exam phases of "EarCog" usage, this module writes down all the information about results in text file, which was then used for building detailed result table (see appendix D).

3.7 Chapter summary

This chapter was fully devoted to the practical side of the thesis, from top-down view on project organization to each module implementation with descriptions of problems encountered and solutions taken.

After the global system design, it was clear, that to implement all needed functionality, four modules where needed to be programmed: module that works with MIDI files and MIDI devices, module with feedback implementations, marker module for examinations and logging module for conducting all analysis during testing phase.

In MIDI module realization, two third party libraries were chosen for utilization: MIDI-dot-NET for working with MIDI input and output devices and NAudio library for parsing MIDI files. During this module implementation, some latency related difficulties were encountered. One was that using WDM driver instead of ASIO driver, there was a latency in receiving MIDI messages — it was solved by using Asio2K virtual driver, which enabled ASIO support for all audio devices. Other problem was that one some processing and calculations were added after receiving signal, latency appeared in the output. This was solved by introducing third thread, which handles all processing and calculation in background and has lower priority, then the one, which works with MIDI input. In this case, no matter how long time takes calculation, it will not affect fast playback on user's input.

Second, interaction providing module, was implemented with help of Accord.NET library. The trickiest part in using Hidden Markov Model was correct configuration of all parameters: states and their initial probabilities, symbols, emission and transition probabilities. To sum up, except obvious notes-states, ghost states where introduced to handle wrong presses, symbols included all possible key presses to process the input, initial probabilities were set in a way, that playing can begin not only from start, although it is less

possible, emission probabilities of ghost states were gradually decreasing with distance from correct note. Finally, transitions from current note not only to the next note, but also to all sequential notes were added for more precise handling missed note case.

Third module, marker, for scoring student's performance in preexams and postexams. For that task, Wikipedia's implementation of Levenstein distance was taken and slightly modified by replacing chars comparisons with Note comparison, by that enabling it to calculate errors count of student's performance. Finally, module, that writes down all examination events to text file was implemented for further testing analysis.

After research and development parts were completed, it was time for testing and evaluation phase to start. Next chapter, "Analysis", is covering it in detail.

Chapter 4

Evaluation and analysis

After completing research as well the design and implementation of "EarCog" project, the phase that is left is testing the hypothesis. This chapter is a full description of all testing phase from tests preparation and setup to the testing process and evaluation of the results.

4.1 Pre-testing

Before testing phase and even before development phase was over, some pre-testing was done to make sure that system was behaving as expected and that the hypothesis at least has a possibility to be proven right.

For pre-testing, three test subjects were tested on a simple task: listen to a broken interval (when two notes are played one after another), and try to figure out both of these notes using two compared techniques — popular among existing solutions and our "deferred judgment feedback".

Although, because it was only pre-testing, and no exact measurements were conducted, from the subjective observation, it was clear that using proposed new feedback correct interval notes were found much faster, that using right/wrong feedback. This subjective observation was by no means any proof of hypothesis, but it was an indication that hypothesis can be positive and that more complete testing can and should be conducted.

4.2 Testing setup

To test the given hypothesis were used:

1. "EarCog" software developed during the project;
2. Set of MIDI files for testing (more on this in subsection 4.3.1);
3. Computer that satisfies the .NET framework 3.5 system requirements;
4. MIDI Keyboard or synthesizer connected to computer;
5. Two randomized groups of students. Students ought to have some basic keyboard playing skills.

4.3 Testing process

Students are randomly separated in two equal-sized groups: A and B. Group A works with usual interaction during learning process, while group B uses "deferred judgment feedback".

The whole process consists of 6 phases.

1. Preexam — both groups students listen twice to the melody. Then they figure it out, by playing without assessment, then they press "SPACE" key for playing for marking. They can repeat last step by pressing "SPACE" again, if they feel they made a mistake (like physically missed the key). They are given two melodies to figure out – 4 notes melody and 8 notes melody. They are always given first note to start with. Score is amount of errors in performed melody. Marking technical side described in section 2.8.
2. Learning — at this phase, both groups students sequentially listen to exercises and try to figure them out. Exercise set described in subsection 4.3.1. This process differs among groups. Group A is interrupted once a mistake is made and bubble-representation of their attempt is shown — green notes are right ones, and red is wrong (see figure 4.1). Group B are never interrupted. Computer follows their play all the time, but they request their results manually by pressing "SPACE" (or results are shown if melody played correct). Same bubble representation is shown, however, in Group's B case it is richer. It shows missed notes, extra notes and

wrong notes — it becomes possible because student is not stopped on first wrong note (see figure 4.2). For both groups, once all notes were figured out, next exercise is switched on.

Figure 4.1: Attempt result example for group A. 5-th note is wrong

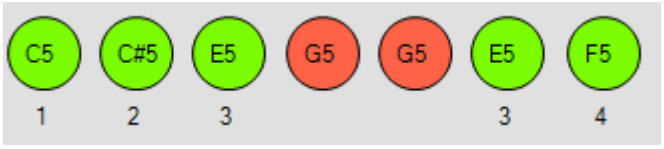

Figure 4.2: Attempt result example for group B. Two notes played wrong. Then 3-rd note played right again; 4-th note played correct.

3. Postexam — Same process, as preexam, but with different MIDI melodies.
4. Compare — Once this mode is selected, feedback is switched, i.e. for group A it is "deferred judgment feedback" and for group B it is classical "instant judgment feedback". Another exercise set from two melodies is loaded.
5. Questionnary answering — Both groups are given a list of questions to fill in. Their answers are for qualitative analysis. A list of questions can be found in appendix C.
6. Retesting after 1 week — Due to specific availability of groups (see "Testing results" for details), it was possible to get same people in a week. At this stage they were retested using same procedure as in preexam and postexam phases to see, whether the effect persists.

4.3.1 Prepared MIDI melodies

For MIDI file preparation, Anvil Studio ™software was used [Willow, 2008]. Anvil Studio is free Windows program for recording, editing, composing and sequencing MIDI music. Such features as track editing and sheet music mode were required for MIDI preparation task and were offered by this program.

The following sets of exercises were made in Anvil Studio (in longer melodies by cutting notes from most likely unknown melodies):

Exams set

- 3 x four notes melodies
- 3 x eight notes melodies

Learning set:

- 2 x two notes melodies
- 3 x three notes melodies
- 5 x five notes melodies (3 used for learning, 2 used for "Compare" phase)
- 3 x eight notes melodies

4.4 Testing results

Group of 18 students of the age 14-16 from summer musical courses were asked to participate in scientific experiment. Children were without sol-fa experience but were experienced at basic musical sheet playing. They were randomly divided into two 9 student groups.

There were two sessions for each group with the interval of 1 week. First musical session included first 5 phases described in section 4.3. Second session's purpose was to see, whether in case of any differences in improvements of group A and group B noticed, the effect persists. It consisted only of phase 6.

4.4.1 Quantitative analysis

The average amount of errors made by both group in all three examinations (before learning, stright away after learning and in a week after learning) is presented in the table 4.1. See all results for each of the participant in appendix D.

	Preexam	Postexam	In a week
Group A. 3 notes task	0.89	0.67	0.75
Group A. 7 notes task	3.44	3.11	3.38
Group B. 3 notes task	1.00	0.67	0.78
Group B. 7 notes task	3.67	2.22	2.33

Table 4.1: Average errors made by group A (control) and group B

Same data for better perception can be represented with percentages taking initial results (received in "preexam" phase) as 100%. It is represented in table 4.2.

	Preexam	Postexam	In a week
Group A. 3 notes task	100%	75.3% (-24.7%)	84.3% (-15.7%)
Group A. 7 notes task	100%	90.4% (-9.6%)	98.3% (-1.7%)
Group B. 3 notes task	100%	67.0% (-33%)	78.0% (-22%)
Group B. 7 notes task	100%	60.5% (-39.5%)	65.5% (-36.5%)

Table 4.2: Error count in percents relatively to preexam results. Group A and Group B.

4.4.1.1 Statistical significance test

To test the results against null hypothesis (i.e. that results could be obtained by chance), Student's t-test for two unpaired samples was applied. This test is used when you have two unrelated (which belongs to different people) arrays of data and you need to verify that difference between them is significant. To do this, at first these arrays were calculated.

First array contained differences in errors count between post- and preexams for all participants. Only long exercise results were taken (7-notes task), because the difference in smaller 3-notes tasks results was not significant. Second one was difference between in-a-week exam and preexam — these differences are smaller, so separated test should be required. Once it was done (look at appendix D for values), p-value (probability of getting such results assuming that null hypothesis is true) was calculated for both of array pairs using online GraphPad calculator [GraphSoftware, 2005]. Then, null hypothesis could be rejected if p-value would be less than significance level, which is taken here as 0.05.

Parameter	Value
Mean	GroupA = 0.33, GroupB = 1.44
SD	GroupA = 0.87, GroupB = 1.01
SEM	GroupA = 0.29, GroupB = 0.34
N	GroupA = 9, GroupB = 9
P value	0.0237

Table 4.3: Unpaired *t*-test calculation results for "postexam minus preexam" dataset

As seen from data represented in table 4.3, P-value for differences between postexam and preexam results equals 0.0237, which makes these results statistically significant. Second *t*-test was run on "in-a-week exam minus preexam" dataset and result is represented in table 4.4.

Parameter	Value
Mean	GroupA = 0.13, GroupB = 1.33
SD	GroupA = 0.64, GroupB = 0.87
SEM	GroupA = 0.23, GroupB = 0.29
N	GroupA = 8, GroupB = 9
P value	0.0056

Table 4.4: Unpaired *t*-test calculation results for "in-a-week minus preexam" dataset

As seen from second *t*-test results, P-value is 0.0056, which means that data can considered to be very statistically significant. Please notice, that there were only 8 participants in in-a-week examination instead of nine — unfortunately 1 participant was absent due to personal reasons.

4.4.2 Qualitative analysis

In addition to quantitative analysis discussed above, also qualitative analysis was conducted. To do this, both groups were asked to complete small questionnaire (see appendix C for whole question list). Answers to questionnaire for both groups were summarized in tables E.1 and E.2 in appendix E.

This analysis was performed to get information about subjects' experience in related skills, like singing and playing by ear) to get their personal opinions on their learning sessions and results as well as some comments for improving software in future. The conclusions from this are primarily anecdotal.

For example, it turned out that all of the participants had an experience at singing in school chorus (they all were from same school), which makes this research more narrow, that expected — and it is still unknown what results would be obtained in group without such experience. Speaking about their improvement feeling, 8 students from 9 in group B reported that they feel an improvement, while only 5 from 9 reported the same from group A. But the most interesting bit of information is in participant's answers on questions "what you disliked in learning session" and "which method you think is better": 6 students from group A reported feeling stressed(scared/nervous) (more thoughts on this in "Future work" chapter) and no one reported the same from group B. However, group B's students complained about absence of immediate feedback which contradicts examination results. Another curious moment is that 7 from 9 group A's student reported that they think group B's method to be more efficient way, however, only 4 from 9 students from group B reported the same.

Chapter 5

Summary and conclusions

5.1 Summary

This work was devoted to the problem of how user interaction is organized in software musical tutors for playing by ear. The need of software tutors for this skill arises because of the complexity in noticing mistakes by your own in the beginning of skill development.

The path of this work started with examining how musical schools deal with this problem. Although it turned out that they do not teach somehow separately to play by ear and this skill develops automatically after long years of studying intervals, this research provided useful understanding of musical dictations which helped in last stage of the work.

Next step was analyzing already existing software solutions to see how this problem is attacked. Common problem was identified in how the interaction is organized: after listening to the melody and student is interrupted on his first mistake while trying to figure it out. Such system obviously has its positives (student always knows when he is wrong) but does not allow to actually use hearing because only one chance is given, so different interaction concept was put forward. Such interaction would allow students to figure out melodies without being interrupted and give them a possibility to check themselves once they need it.

The theoretical part of the work continued with choosing appropriate algorithms for future implementation. Among Levenstein distance, sequence alignment and Hidden Markov models the latter were chosen for score following and the first one was chosen for musical exams marking implementation. Also, a light study on MIDI format was done.

Once all theoretical background was finished, practical part was started. After top-level design and choosing development tools, author implemented module for working with MIDI devices and reading MIDI files with help of third party components and some special program for emulating ASIO driver to reduce sound latency. Later on, Hidden Markov models module was developed through configuring each of HMM parameters and consulting existing works on HMM application in score following area. Finally, marker module implementation of Levenstein distance and logging module were completed. Once all components were ready, checking the hypothesis by conducting an experiment was left.

To test proposed interaction against commonly used, 18 students of musical courses were randomly divided into two equally sized groups. Both groups were given same exam and were marked in a same way. After that group A worked with commonly used interaction while group B worked with proposed one. The learning sessions lasted about one hour. Once finished, both groups were given another exam to measure their improvements. Each exam consisted of short and long melodies. While differences in results on short melodies tasks were insignificant, group B performed much better on longer melody, then group A: on average, students from group B made 39.5% less errors than on initial exam (NB: all three exams were on different melodies, so it does not mean anything; matters only difference in groups results), while group A students made only 9.6% less errors than on initial exam (for full results table, see appendix D).

After that students were asked to answer a few questions about learning session. It turned out that group A became more nervous and annoyed during the learning phase than group B and this could be the reason of such differences in their results. To make sure that the effect maintains after some time same test subjects (except 1 student) were tested again in a week and once again results were compared with initial exam ones. This difference was a bit smaller, but same tendency was seen, i.e. group B mistakes count was 36.5% less than on first exam while group A only 1.7% less.

To make sure that results could not be obtained just by chance, statistical significance test using Student's t-test was run. Both improvements detected after learning session and one week after turned out to be statistically significant.

One interesting thing noticed in questionnaire answers was that while group B were clearly showing better results and never complained about being scared, less people from this groups said that proposed interaction method was more convenient, that students from group A (after second exam there was a time for both groups to change another

method for comparison).

5.2 Conclusions

Conducting researches, implementing software and running tests on subjects were not the aim of this work. Rather, these all were tasks and tools to achieve main target: prove that hypothesis is true or that it is wrong. The hypothesis stated (see section 2.3 for formal definition) that by allowing students to figure out the melody by themselves and assess them only when they want it, the development of playing by ear skill is accelerating; results of testing showed that this is truth and therefore hypothesis is confirmed.

However, the experiment was not perfect. First of all, it was rather limited in both human and time resources. Although results were proven to be statistically significant, such limitations can cause not clear results any way. Second, even if on larger set of test subjects and longer learning periods the results would be the same, the reasons for it might be different. One of the possible issues which author is concerned about is that students from first group reported feeling stressed because of instant interruptions. Although some students from group B mentioned that absence of instant mistake correct is disadvantage, in reality group A students because of feeling stressed might also had felt less motivation and discouraged to endeavor. In this case positive results could be explained not by ability to use there hearing as it was supposed, but simply by not being stressed. Also, the reason of observed benefit may be both detailed feedback and absence of interruptions, or only one of them. These and other potential research problems are described in chapter 6, “Future work”.

To summarize, the results achieved indeed indicate that proposed method is *potentially* not only more efficient but as well more comfortable way to organize software playing by ear tutors and this is a good start. But to verify this, more extensive research should be conducted.

Chapter 6

Future work

6.1 More extensive testing

Although the results of testing are very promising and make solid indication that improved interaction is indeed more efficient way to develop skill of playing by ear, it is still not a guaranteed valid proof. There are two major reasons for this: not extensive enough testing and unchecked cause of effects.

Due to the limited amount of time as well as human resources, this work's testing is not sufficiently extensive. First of all, there were only 18 available students in total, 9 in each group. Although analysis showed, that results are indeed statistically significant, such research is by all means too narrow. For example, the age, as well as skills of participants were approximately the same, and, while it was good for this testing, we have no information of how students with different (like without even basic musical education) skills would benefit from provided way of learning. Second, participants has only one session, which was around 1 hour. There is no scientific evidence, that, for example, both group's improvements would not equalize after a few more sessions, in other words, that observed skill development acceleration was not temporary.

Unchecked cause of effects are the second potential weak part of the analysis. Due to different possibilities provided by different methods of learning, the visual information in "deferred judgment feedback" mode was also richer: by using HMM and not interrupting student at all, "EarCog" program was able to show, that, say, note number two was skipped, but notes after that were played correctly; "instant judgment feedback" was

stopping student on first made mistake, which limits the among of information that student could possible get. So, although we have seen increased improvement in this group, we are still not sure, that this was exactly giving time to figure out on his own, that gave the benefit. And, while such possibility might seem unlikely, the other possible problem with causes of benefit is more serious — as qualitative analysis of student's answers to questionnary showed, group A, who was interrupted on their mistakes were becoming more and more nervous during the learning procedure and felt fear of being wrong and of necessity to start new attempt. This fear could (and probably did) influence their attitude to learning process itself and lower their motivation.

So, the future work related to all this issues would be to provide more extensive research with more groups of different skills as well as with more participants in each group. Also, it would be helpful to change the way visual feedback is presented and measure the differences in improvement again — for example, it might be best for "clearer" results to reduce visual feedback for group B to see, whether this particular part of interaction causes an improvement. And also, increasing learning sessions count is wise for checking, whether the improvement is only on first stages of learning, or it persists during all phases, or even it accelerates.

Another thing is the length and difficulty of melodies for learning. There was a clear indication, than improvement is seen on longer melodies, but its unknown whether it's true for any lengths. Also, only 1-note "chords" were supported by "EarCog". It would be helpful to increase program's functionality by supporting chords (both MIDI support and HMM support), and test, how difference in improvements changes during learning melodies with chords (see section 6.2).

Last thing would be type of measurements per se. Due to time limited resources, only pitches were tested in "EarCog". However, there are also two other parameters, which together with pitches makes the music — its velocity and rhythm (see section 6.2). Examining these two characteristics would make the research more thorough.

6.2 More advanced functionality

As already mentioned above, only pitches and one-note-chords where supported in "EarCog" project, because these are most basic "layer" of playing by ear skill.

However, to make this program more complete and usable for real world application, rhythm and velocity measurements should also be included. It would probably be better to allow students to test all these parameters separately, before trying to achieve excellence in all three areas. Both rhythm and velocity could be incorporated in Hidden Markov models to allow student free from distraction learning on his own, yet providing full information about what was played correctly and what was not.

Even more intriguing seems including support of chords, because detecting the right chords is much challenging, than only one note at a time.

6.3 Implementing learning in HMM

One of the tasks that can be achieved by Hidden Markov models is learning. Learning in HMM is a process where a set of output sequences is taken and given probabilities are modified, so that the likelihood of getting such output sequence is highest. Nowadays, Baum-Welch as well as Baldi-Chauvin algorithms are used for this purpose [Baggenstoss, 2000].

For current project there are at least two potential applications of HMM learning possibility. First one is to use learning to adjust student's model to real user and increase quality of score following. Second is to use this adaptation of student's model to extract useful information from it and make judgments about user's personal weak spots in music and optimize learner's path [Beck and Woolf, 1998] [Murray, 1998].

6.4 Adaptation to singing and humming by ear

Current work tackles the playing on MIDI device by ear. How this could be logically developed in future is singing and humming by ear, which is very intriguing. This functionality would allow students to develop their vocal abilities. An extension like this would also probably include measuring, how stable each performed note is as well as an ability to perform smooth transition from one note to another.

There are good tools to extract pitch from sung output, but the implementation would need to deal with pitch quantization techniques to decide which real pitch "maps" with which sung pitch.

Appendix A

"EarCog" interface screenshot

Figure A.1: Main window screenshot. Properties window is hidden and accessible only by CTRL+F3 press. For third exam in a week "Preexam" mode was used.

Appendix B

Configuration instructions

Configuring "EarCog" can be done through "Properties" window displayed on figure B.1, which can be accessed by "CTRL+F3" hotkey, so that test subjects would not unintentionally access it. Configuration should be completed before first use.

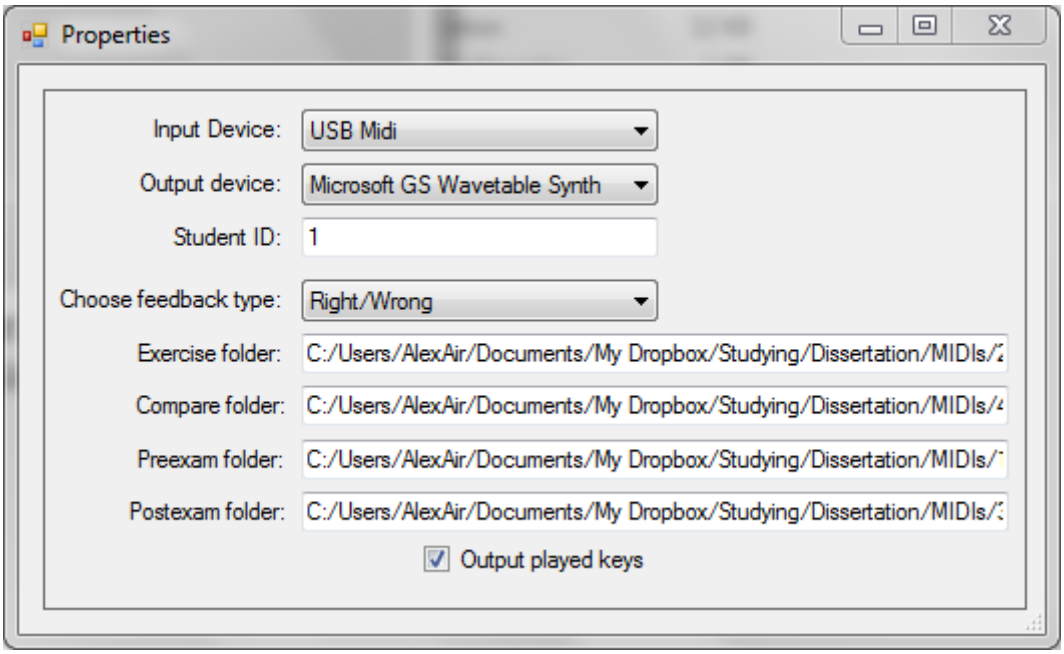

Figure B.1: Properties window of "EarCog". Can only be opened by "CTRL+F3" hotkey

All settings are mandatory, and they are:

1. `Input device` — MIDI device, should be chosen from drop down. If device is connected after program was opened, program should be restarted.

2. `Output device` — Output device for playing back the input.
3. `Student ID` — Identifier of student, used by logger.
4. `Feedback type` — "Right/Wrong" of "Deferred" should be chosen from the drop down list.
5. `Exercise folder` — folder, that contains MIDI exercises used for second phase of testing process — learning. Files alphabetical order should be match the order of planned loading.
6. `Compare folder` — folder, that contains MIDI exercises used for "Compare" phase of testing process. Same order rule applies.
7. `Preexam folder` — files from this folder are loaded for "Preexam" mode. Same order rule applies.
8. `Postexam folder` — files from this folder are loaded for "Postexam" mode. Same order rule applies.
9. `Output played keys` — should be checked, if MIDI device doesn't have build-in speakers. It is often a case, if MIDI-keyboard is used, and rarely, when synthesizer is used.

All settings are saved once the window is closed.

Appendix C

Questionnaire for participants

Due to the fact, that participant's native language was Russian, the questionnaire they were given was also written in Russian language. Both translated to English and original versions are presented below.

English:

1. Do you have any experience in playing by ear? Describe it.
2. Do you have any experience in vocal trainings? Describe it.
3. How much, you feel, you have improved your playing by ear skills? (very improved/improved/not improved/tired and performed worse)
4. Positive moments during learning?
5. Negative moments during learning?
6. How helpful was learning in total? (very helpful/helpful/not very helpful/not at all)
7. What did you think about other technique tried in "COMPARE" mode? Was it more/less useful? Why?

Russian:

Имя/Фамилия _____/_____

1. Есть ли у Вас опыт в игре на слух? Опишите его.
2. Есть ли у Вас опыт в изучении вокальной техники? Опишите его.
3. Помогла ли Вам предоставленная программа сегодня в обучении?(очень помогла/помогла/не особо помогла/совершенно не помогла)
4. Как Вы считаете, насколько Вы сегодня улучшили свои навыки в игре на слух?(очень улучшил(а)/улучшил(а)/не особо улучшил(а)/не улучшил)
5. Опишите положительные моменты в сегодняшнем обучении?
6. Опишите негативные моменты в сегодняшнем обучении?
7. Что Вы думаете о втором методе, с которым ознакомились в режиме «СРАВНЕНИЕ»? Был ли он более/менее удобный? Почему?

Appendix D

Detailed testing results

	Group A (CONTROL)								
Student id:	1	2	3	4	5	10	11	12	13
Day:	Sat	Sat	Sat	Sat	Sat	Sun	Sun	Sun	Sun
Before training									
Errors (Task 1, 3 notes)	1	1	0	0	1	2	1	2	0
Errors (Task 2, 7 notes)	3	4	3	1	4	4	4	5	3
After training									
Errors (Task 3, 3 notes)	0	1	0	1	0	2	1	1	0
Errors (Task 4, 7 notes)	4	3	3	0	3	4	3	4	4
In a week									
Errors (Task 5, 3 notes)	1	1	-	0	1	1	1	0	1
Errors (Task 6, 7 notes)	3	4	-	2	4	4	3	4	3
Postexam - preexam (for t-test)	-1	1	0	1	1	0	1	1	-1
In a week - preexam (for t-test)	0	0	-	-1	0	0	1	1	0

Figure D.1: Group 1 participants' results for all tasks

	Group B								
Student id:	6	7	8	9	14	15	16	17	18
Day:	Sat	Sat	Sat	Sat	Sun	Sun	Sun	Sun	Sun
Before training									
Errors (Task 1, 3 notes)	0	1	1	1	2	0	1	2	1
Errors (Task 2, 7 notes)	2	3	4	4	5	4	3	4	4
After training									
Errors (Task 3, 3 notes)	0	1	1	1	0	1	1	1	0
Errors (Task 4, 7 notes)	1	2	1	2	3	2	3	4	2
In a week									
Errors (Task 5, 3 notes)	0	1	0	1	2	1	1	0	1
Errors (Task 6, 7 notes)	1	2	2	1	4	3	3	3	2
Postexam - preexam (for t-test)	1	1	3	2	2	2	0	0	2
In a week - preexam (for t-test)	1	1	2	3	1	1	0	1	2

Figure D.2: Group 2 participants' results for all tasks

Appendix E

Questionnaire results table

Parameter	**Group A**
Experience in playing by ear	6 students reported trying to play by ear at home
Experience in singing	Everyone sang in chorus at school
Feeling of improvement	5 students felt improvement
Disliked in learning	6 students reported feeling highly stressed
Comparing with method B	7 said it's better

Table E.1: Summary from questionnary reports received from test subjects. Group A

Parameter	**Group B**
Experience in playing by ear	5 students reported trying to play by ear at home
Experience in singing	Everyone sang in chorus at school
Feeling of improvement	8 students felt improvement
Disliked in learning	4 students disliked absence of immediate indication
Comparing with method A	4 said it's better

Table E.2: Summary from questionnary reports received from test subjects. Group B

Bibliography

[ActiveMusician, 2011] ActiveMusician (2011). Active musician. ars nova practica musica. http://www.activemusician.com/item--TW.ARSPM42; Last accessed: 14/06/2011.

[Adventus, 2011] Adventus (2011). Ear training coach. adventus. http://www.adventus.com/store/ear-training-coach/; Last accessed: 14/06/2011.

[Baggenstoss, 2000] Baggenstoss, P. M. (2000). A modified Baum-Welch algorithm for Hidden Markov models with multiple observation spaces. In *Proceedings of the Acoustics, Speech, and Signal Processing, 2000. on IEEE International Conference - Volume 02*, ICASSP '00, pages II717–II720, Washington, DC, USA. IEEE Computer Society.

[Bard, 2007] Bard, G. V. (2007). Spelling-error tolerant, order-independent pass-phrases via the Damerau-Levenshtein string-edit distance metric. In *Proceedings of the fifth Australasian symposium on ACSW frontiers - Volume 68*, ACSW '07, pages 117–124, Darlinghurst, Australia, Australia. Australian Computer Society, Inc.

[Beck and Woolf, 1998] Beck, J. E. and Woolf, B. P. (1998). Using a learning agent with a student model. In *Proceedings of the 4th International Conference on Intelligent Tutoring Systems*, ITS '98, pages 6–15, London, UK. Springer-Verlag.

[Chimbombi, 2007] Chimbombi, O. (2007). *Basic Tonic Solfa Concepts: Your Easy Guide for the Best Choral Training and Performance*. Number ISBN-1434301648. AuthorHouse.

[Chu and Li, 2011] Chu, W.-T. and Li, M.-L. (2011). Score following and retrieval based on chroma and octave representation. In *Proceedings of the 17th international conference on Advances in multimedia modeling - Volume Part I*, MMM'11, pages 229–239, Berlin, Heidelberg. Springer-Verlag.

[Draper, 2005] Draper, S. (2005). Feedback. http://www.psy.gla.ac.uk/ steve/feedback.html; Last accessed: 12/07/2011.

[Erichsen, 2002] Erichsen, T. (2002). Asio2ks - generic asio driver for wdm soundcards. http://www.asio2ks.de/; Last accessed: 09/07/2011.

[Forney, 1973] Forney, D. (1973). The viterbi algorithm. In *Proceedings of the IEEE.*

[Gales and Young, 2007] Gales, M. and Young, S. (2007). The application of Hidden Markov models in speech recognition. *Found. Trends Signal Process.*, 1:195–304.

[GraphSoftware, 2005] GraphSoftware (2005). Graphpad quickcalcs: t-test calculator. http://www.graphpad.com/quickcalcs/ttest1.cfm; Last accessed: 10/08/2011.

[Havvej, 2011] Havvej, E. (2011). Earmaster pro 5.0 software for aural test practice. http://www.earmaster.com/; Last accessed: 14/06/2011.

[Heath, 2011] Heath, M. (2011). Naudio. http://naudio.codeplex.com/; Last accessed: 09/07/2011.

[Jordanous and Smaill, 2008] Jordanous, A. and Smaill, A. (2008). Artificially intelligent accompaniment using hidden markov models to model musical structure. In *Proceedings of the fourth Conference on Interdisciplinary Musicology (CIM08).*

[Kim and Kececioglu, 2008] Kim, E. and Kececioglu, J. (2008). Learning scoring schemes for sequence alignment from partial examples. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 5:546–556.

[Lokovic, 2009] Lokovic, T. (2009). midi-dot-net – a library for midi programming in c#. http://code.google.com/p/midi-dot-net/; Last accessed: 01/07/2011.

[Mayers, 2011] Mayers, R. (2011). Play by ear. http://www.iwasdoingallright.com/playbyear/; Last accessed: 14/06/2011.

[Miller et al., 2009] Miller, F. P., Vandome, A. F., and McBrewster, J. (2009). *Levenshtein Distance: Information theory, Computer science, String (computer science), String metric, Damerau?Levenshtein distance, Spell checker, Hamming distance.* Alpha Press.

[Murray, 1998] Murray, W. R. (1998). A practical approach to bayesian student modeling. In *Proceedings of the 4th International Conference on Intelligent Tutoring Systems*, ITS '98, pages 424–433, London, UK. Springer-Verlag.

[Orio and D'echelle, 2001] Orio, N. and D'echelle, F. (2001). Score following using spectral analysis and Hidden Markov Models. In *Proceedings of the 2001 ICMC, Cuba.*

[Orio et al., 2003] Orio, N., Lemouton, S., and Schwarz, D. (2003). Score following: state of the art and new developments. In *Proceedings of the 2003 conference on New interfaces for musical expression*, NIME '03, pages 36–41, Singapore, Singapore. National University of Singapore.

[Pardo, 2005] Pardo, B. A. (2005). *Probabilistic sequence alignment methods for on-line score following of music performances.* PhD thesis, Ann Arbor, MI, USA. AAI3163905.

[Philpott, 1996] Philpott, R. (1996). Virginia tech multimedia music dictionary. `http://www.music.vt.edu/musicdictionary/`; Last accessed: 22/07/2011.

[Riben, 2011] Riben, P. (2011). Ear training happy note. `http://www.happynote.com/ear-training.html`; Last accessed: 14/06/2011.

[Richter, 2006] Richter, J. (2006). *CLR via C#, Second Edition.* Number ISBN-13: 978-0735621633. Microsoft Press; 2nd ed. edition (March 22, 2006).

[Ruska, 2011] Ruska, J. (2011). Trainear. `http://www.trainear.com/`; Last accessed: 14/06/2011.

[Schoeberl, 2011] Schoeberl, M. (2011). Good ear. `http://www.good-ear.com/`; Last accessed: 14/06/2011.

[Souza, 2010] Souza, C. (2010). Hidden markov models in c#. `http://www.codeproject.com/Articles/69647/Hidden-Markov-Models-in-Csharp.aspx`; Last accessed: 16/07/2011.

[Stamp, 2004] Stamp, M. (2004). A revealing introduction to Hidden Markov models.

[White, 2000] White, P. (2000). *Basic MIDI.* Number ASIN: 1860742629. Sanctuary Publishing Ltd.

[Wikipedia, 2011] Wikipedia (2011). Levenshtein distance. `http://en.wikipedia.org/wiki/Levenshtein_distance`; Last accessed: 01/08/2011.

[Willow, 2008] Willow (2008). Anvil studio ^{TM}catalog. `www.anvilstudio.com`; Last accessed: 16/07/2011.