BSc Thesis

ELSET:
e-Learning for Solfege and Ear Training

MMI-BS-2004-03

Graduation Committee:
Advisors: M. Pantic, TU Delft
Advisor: I. Vujovic, Royal Conservatory
Member: C.A.P.G. van der Mast, TU Delft

BSc Candidates:
Rachid Gangadajal
INTRODUCTION ......................................................................................................................... 2
CHAPTER 1: ANALYSIS OF ‘PRACTICA MUSICA’ ................................................................. 3
CHAPTER 2: PROBLEM DEFINITION & REQUIREMENTS ...................................................... 4
CHAPTER 3: OVERVIEW OF THE TOOLS USED ................................................................. 5
CHAPTER 4: SYSTEM DESIGN ............................................................................................. 6
  4.1 Functional Specification ............................................................................................... 6
  4.2 Implementation ........................................................................................................... 10
  4.3 Classes ....................................................................................................................... 16
  4.4 Exercises of the ELSET System ............................................................................... 19
CHAPTER 5: PROGRAM EVALUATION .............................................................................. 25
CHAPTER 6: CONCLUSION & FUTURE WORK ................................................................. 26
APPENDIX A: PROCESS REPORT .................................................................................... 27
APPENDIX B: THE USER’S MANUAL FOR ELSET ............................................................ 29
Introduction

For my Integrated Project at the University of Delft the topic has been defined by the Royal Conservatory at The Hague. The Conservatory is one of the most renomated schools of The Netherlands when it comes to teaching of music. A lot of students from all over The Netherlands come to The Hague for a musical study and/or to practice/improve their musical skills.

In the late 1980’s half of all workers in the Netherlands were using computers. The school system followed this trend: the proportion of schools with computers rose from 18% to 92%. And as computer technology became more advanced, so did the educational software that was marketed to schools. Computer-assisted Education (CAE) was, and still is, promoted as a potentially revolutionary new means of learning. PCs and general applications software made computing more flexible and easily adapted to different subjects and styles of teaching. The explosive growth of the Internet and World Wide Web only reinvigorated the use of computers in classrooms.

Many projects and exercises can or should be done by means of a personal computer, either at home or at school. Another phenomenon related to computers is web-lecture: the lecture that is normally given in classrooms can now be followed over the Internet. But due to the fact that not all students are assumed to have internet-access, much educational software is being stored on CD-ROM.

Another motivation to do so is the fact that nowadays students are required to do more on their own or in collaboration with fellow students.

The goal of this project is to implement such a software package for first year students of the Conservatory that can be stored on CD-ROM. The intention of such a program is not mainly focussed on lectures, but more on learn-by-doing exercises. The student will then have the possibility to train their skills, as if there is a teacher present telling them how to do it.

The Conservatory is currently using a software package called “Practica Musica”, but it is not entirely what the Conservatory have in mind for the students. There are a lot of features that can or should be improved to meet the requirements. Also not all the wishes of the teaching group are supported by this package.

Before I continue with specifying the requirements, I’ll first give an overview of the analysis I have done on Practica Musica. Based on this analysis I will specify in chapter 2 the requirements that the new tool for computer-assisted (e-learning) solfege and ear training (ELSET) should fulfil. Next I will explain the tools I have used to realise this project in chapter 3. Chapter 4 covers the system design, while chapter 5 will discuss the evaluation of the program. Finally, in chapter 6, I will give an outline of future work that can be done. Appendix A includes a short summary of the process, while Appendix B contains the user manual for the ELSET system.

For this project I had contact with Ida Vujovic regularly, who is a teacher at the Conservatory. She was in charge of creating the exercises, which I implemented in the ELSET system. Since I don’t have musical experience and knowledge, she was providing me with the necessary.
Chapter 1: Analysis of ‘Practica Musica’.

There are many packages/programs which enable one to practice their musical skills, for example ‘EarMaster’, ‘GNU Solfege’, ‘Perfect Pitch’ & ‘PianoPlayer+’. All these musical programs are written to help one practice ear training. Some of the exercises that can be done with these kinds of packages are:

- Recognise melodic and harmonic intervals
- Sing the intervals the computer asks for
- Identify chords
- Dictation
- Remembering rhythmic patterns

Another one of these programs is Practica Musica, which is currently being used by the Royal Conservatory. Practica Musica is a complete music theory and ear training package, it contains selectable skill levels to adjust the difficulty to suit students from beginner to advanced. An on-screen keyboard may be used for note entry.

This program served as a good example for the new one I was supposed to develop. An analysis of this program answered a lot of my questions about the requirements. During the analysis of Practica Musica I looked for features that could be useful for the new program and others that can be improved. The results are listed below:

What could be improved in Practica Musica?

- You have to make your own decision about what exercise you’re going to do, choose an activity and level.
- You first have to set the sound settings correctly, or else the program will not work.
- Feedback to the user is done in a small white area.
- There is no straightforward interface: every activity has another interface with other functionality for the buttons, so it is not clear what to do exactly within a certain activity.
- In the main screen everything is possible, so you start in a screen without any clue what to do exactly.
- Music-notation bars are not automatically set on the music stave, the user has to do this himself on the right place.
- The program works with its own music formats; there is no possibility for importing other music formats such as MIDI-files.
- The program is mainly focussed on making exercises instead teaching the theory of music. Moreover there is no pre-described way a user can follow for completing the exercises.

The items summed up above will be focussed on during the design of the ELSET system, as described in Chapter 4.

What could be useful?

- To have an option “Play my notes” for playing the user input on the music stave.
- To have the option for placing notes on the bar go as follows: First choose the desired note you want to place and then click on the right place on the bar.
- By means of boxes around wrong notes the user can see which mistakes he made.

The above are features that are currently available in Practica Musica, and that we also wish to include in the ELSET system. As you can see in Chapter 4, these items are all taken into consideration during the design.
Chapter 2: Problem definition & Requirements

Since none of the programs mentioned in the previous chapter contains all the requirements and features the Conservatory demands, they decided to have their own program developed, containing all the features and requirements according to their method of teaching. For example, as we saw in the previous chapter, Practica Musica was not able to import MIDI-files, while the Conservatory works a lot with them. By developing their own software package, they would be able to integrate all their wishes, which wouldn’t be possible with the other existing programs.

Let’s start by stating the definition of the project given by the Conservatory:

*This tool is aimed at computer-aided training of aural skills and should enable the students to do their (almost entire) practical training at home. It should represent a program for independent learning.*

*The software package has been envisioned as a set of different kinds of exercises for aural skills training. The main tools that are to be used by these exercises are a piano and a music-stave.*

*While the teacher will not be present at the time, the student must get some real teaching.*

The requirements for the ELSET system can be divided in pedagogical and system requirements. The pedagogical requirements explain how the ELSET system should behave as an educational tool, while the system requirements will specify the functionalities and features the system should include.

**Pedagogical requirements:**
- First of all, the program must be suitable for first year students of the Royal Conservatory of The Hague.
- The program should offer the most important possibilities for students to practice their (theoretical and practical) music skills without teacher’s assistance.
- The program should have a form of guidance for the student.
- Level of difficulty should be increased as student gets further in the program.
- The student must be provided with the theory needed to perform the exercises belonging to some specific section.
- The exercises must be of different kinds, divided in lessons belonging to a certain theory.
- The student must get appropriate feedback when giving answers and performing the tasks.

**System requirements:**
- The program should contain a piano.
- The program should contain a music-stave that is connected in some way to the piano.
- The program must be able to use music files in MIDI and MP3 format.
- Playing notes on the piano should automatically place the corresponding notes on the correct place on the music-stave.
- The appropriate feedback must be shown to the user while performing the tasks.

As explained in the rest of this report, the ELSET system fulfills all the above listed requirements. In Chapter 4 you can find descriptions of the design for all system requirements while taking the pedagogical ones into consideration.
Chapter 3: Overview of the tools used

A first step in the development of the ELSET system was to define the programming environment. I did so by choosing the following:

**Java, Visual Café 4.0 & Standard Development Kit (SDK) 1.4.1**

The implementation of the ELSET system was done in Java, using the programming environment Visual Café 4.0 from Symantec. The reason why I chose this programming language is because of Java’s platform independency. Since each student should be able to use ELSET anywhere on whatever system (UNIX, Windows,...), Java seemed the best option. Integration with websites can be also easily done, if a program is written in Java. Another motivation for my choice is based on the simple sound features that Java offers. Since sound is a crucial feature of the program it is important that I’m able to implement this in the best way possible. Visual Café is the programming environment I have used when learning Java; I have a great experience using this program for implementing user-interfaces with Java Swing objects. The Visual Café 4.0 environment does not include the latest SDK-package; this package is needed for the newest Java methods and procedures. At the time I started programming, the last version of the Standard Development Kit was 1.4.1. Any newer version of this will be OK, but certainly not older ones. Version 1.4.1 contains the javax.sound class, which offers procedures for accessing MIDI-files and other musical attributes. The SDK 1.4.1 can be downloaded from the Java Sun website ([http://sun.java.com](http://sun.java.com)).

**jMusic**

Another important package I have used is called jMusic. jMusic is a programming library designed by Andrew Sorensen and Andrew Brown to provide composers and software developers with a library of compositional and audio processing tools. It provides a solid framework for computer-assisted composition in Java, and is also used for generative music, instrument building, interactive performance, and music analysis. jMusic supports musicians with its familiar music data structure based upon note/sound events, and provides methods for organising, manipulating and analysing that musical data. This package provides useful musical features for midi-files and contains a music-stave, which can be accessed and edited. It has a full online documentation and represents the main function-provider for the ELSET system. There are more packages like jMusic, for example JFugue ([http://www.jfugue.org/](http://www.jfugue.org/)) and NoSuch Midi ([http://www.nosuch.com/nosuchmidi/](http://www.nosuch.com/nosuchmidi/)). However, I chose jMusic because it is a free and open-source package. As it already contained most of the basic functionalities, I could easily upgrade the classes with self-programmed functionalities. For more information on jMusic, see: [http://jmusic.ci.qut.edu.au/](http://jmusic.ci.qut.edu.au/)

**NativeJ**

The Java-program build in Visual Café can’t be run independently on a computer. NativeJ is a free software-package that enables one to compile his or hers Java- and/or Jar-files into one executable file that can be started up independently in a Windows console. By including all the classes and packages used to develop the Java program in Visual Café in NativeJ, all these files are compiled into a stand-alone program for Windows having an ‘.exe’ extension.
Chapter 4: System Design

After having discussed the requirements and possible solutions with Ida Vujovic (as explained in Chapter 2) we had to make some practical decisions on how the interface should look like, how navigation through program should be done and what should be visible when doing exercises. The following was decided:

1. **Applet or application**: Based on the requirements of the Royal Conservatory for the program, a decision has to be made whether to build a Java applet or a Java application. Therefore let’s take a look at the pros and cons of these two different types of implementation. First of all, an applet cannot access the local file systems and thus cannot open, save or delete files. This problem can be solved by defining additional parameters on the computer on which the program should be installed. In any case the option of accessing the file system must be present, because the program should be able to open and/or read midi-files. Further, with an applet it is impossible to have native calls and therefore it is impossible to access the local hardware. The latter is necessary for connecting to the sound device of the computer. On the other hand, it is not possible to run an application on the Web. But this feature is not really important because the program is being built for a CD-rom. So the choice was easily made based on these considerations: building ELSET as a Java application seems to be the best choice.

2. **Navigation** must be kept as simple as possible. It is not a task of the user to discover how to navigate through the program; the main activity of the program is to train one’s musical skills. Implementing navigation in a standard way is therefore desirable. Menu’s and next/previous-buttons should be able to accomplish this.

3. **Feedback** to the user must always be visible at the same place in the window, to omit confusions.

4. The most important feature of the program, that is the connection between the virtual piano and the music-stave, must be implemented in such a way that when playing notes on the piano the corresponding notes appear on the music stave.

This chapter will give an overall overview of the program design, its functional specification. Three ELSET exercises will be also described in terms of command flows.

### 4.1 Functional Specification

ELSET system contains different exercises with a certain level of difficulty and all these exercises (thus) have different functionalities. So the program can be seen as a collection of functionalities which are summarized in this paragraph. As I divided the requirements in Chapter 2 in pedagogical and system requirements, I will also separate the functionalities according to these terms.
Pedagogical functionalities:

a) The program is designed according to the example-based (learn-by-doing) teaching paradigm.

The program is intended to provide the user with practical examples and exercises to train his musical skills. As the level of difficulty increases with each lesson and the user is offered enough practice opportunities, his musical skills and knowledge is improved. In this way the learn-by-examples teaching style is met.

b) The user can practice all exercises on his own, without the teacher’s assistance.

As the theory is explained for every exercise and there is extra help available, the user should be able to complete every exercise on his own, without any external help from for example the teacher. The program also provides the user with the necessary feedback and, together with the theory and extra help, the program contains all the ingredients to serve as a tool for students, with which they can practice entirely on their own.

System functionalities:

a) The user can easily play some melody on the piano.

This is one of the features that is available in all of the packages of this kind. It enables the user to play on the (virtual) piano just for practice, independent of the exercise he or she is doing. The notes played by the student are inserted automatically on the music stave and can be played back repeatedly. So the user can see what he or she is playing. The user can play on the piano by clicking on the keys with his mouse. Implementation of the piano is explained in section 4.2.

b) The user is able to insert some notes on the music-stave and let the program play the melody.

Besides the piano the user is provided with another tool for inserting notes on the music stave. This is required by some of the exercises, where the user needs to insert a certain melody on the music stave and the program evaluates it. To accomplish this the program shows the user a set of different notes he can insert. By choosing the desired note and clicking on the right place on the music stave, that ‘note’ is inserted and the user can playback the sequence he inserted so far. Notes can be deleted and inserted repeatedly. Implementation of the music-stave is explained in section 4.2.

c) The program is able to extract several data from MIDI-files for different purposes.

The jMusic-package contains a lot of methods to extract data from a MIDI-file, which turned out to be very useful in the implementation of the exercises. Not only data from a MIDI-file itself, but also from the stave can be extracted by using jMusic-calls. For example, in many exercises it is required that the user can listen to some MIDI-file and should see the music-notation of that file on the music-stave. This can easily be done in the following way:

To put the music-notation of a file on the music-stave you need a “Phrase” of that file. A Phrase is a sequence of notes. But when reading a MIDI-file, it is not possible to extract that Phrase at once. jMusic contains the procedures readScore(), getPart() and getPhrase() which will lead to the Phrase.
First read the MIDI-file into the Score, get the desired Part of that score, and make a Phrase of it which can be put on the music-stave.
Getting information from a Phrase is also supported by jMusic, for example you can get “pitch numbers” from the notes, which relates to the midi numbers from a keyboard.

d) The user is able to compare his melody with some other existing melody. By either playing on the piano or by inserting single notes on the music stave, the student can compose a melody. In many exercises this melody needs to be evaluated against the correct answer that is stored in a file. Most of the time this file is a MIDI-file. Evaluation is then possible by extracting the ‘Phrase’ of that MIDI-file. A Phrase a sequence of notes. The Phrase of the melody on the stave can also be extracted and evaluation is then done as follows:

The systems compares these two Phrases note by note. After two notes are compared, they can either be identical or not. In case they are identical (when the inserted note is correct) the note is given a color attribute ‘black’, and this note is added to a new Phrase. If the inserted note is not correct the note is coloured ‘red’ and also this one is added to the new Phrase. After all the notes of the phrase inserted by the user are evaluated, the new Phrase with black and/or red notes is put on a new music-stave and shown to the user.

For more information on this evaluation, see section 4.3 and section 4.4 exercise 1.

e) The program is able to load and play a MP3-file.

Some exercises require the user to listen to some piece of music before he can continue with the rest of the exercise. Some of these files to listen to are in midi-format, which can be played using the javax-routines. But some files are also in MP3-format, which Java doesn’t support.

Using an extra package, called Java Layer, it is possible to convert the MP3-file to a standard WAVE-file, which is in turn passed through to Java’s Midi Synthesizer. From here, the implementation is the same as the one for playing Midi-files.

f) The user is able to load a music file in MIDI or MP3 format and play it as many times as desired.

One of the requirements from the Royal Conservatory is that the program must be able to import self-made MIDI or MP3 files. By using the sound-device of the Java-package a file can be read and the tones to be played are then sent to the soundcard of the computer. The program offers the possibility to repeat this piece of music as many times as desired. See section 4.4 for more information on the implementation of this.

g) The program is able to play a random file from a collection of files.

Another feature that is required for some exercises is the playing of MIDI-files or MP3-files at random. For this, two new java-classes were made, one for MIDI and one for MP3. In these classes a random number-generator is used to choose a random number. The implementation of this is as follows:

By means of the random number-generator a number between 0 and the number of files in the collection is chosen. Per random number it can be specified which file should be played, what the settings for the music-stave are, and what the evaluation data are.

When one of the file-collections is being called by the exercise-procedure, the entire class is scanned for the current exercise-number wherefrom the settings can be found on how to proceed.
h) The user is able to load a particular melody in the music stave from a file. As described in d) the user can import MIDI or MP3 files. In some of the exercises it is required to show that melody on the music stave as well. By reading the Phrase of the file the notes can be inserted one by one on the music stave. How this is done, has been described in item c.

i) The user can answer the questions of an exercise and then get an appropriate feedback.
In every exercise the user has to accomplish one or more tasks. These tasks consider either answering some questions or playing/listening to a certain melody and taking some actions after that. In any case, the program evaluates the user’s answer and returns a proper feedback. For example, in the case of giving answers by playing a melody, and after the input is evaluated, the program shows which notes are correct and which are not by showing them in different colours. In the GUI implementation (section 4.2) the different kinds of feedback are described.

j) The user is able to get additional information on any subject.
In some cases a certain theory is explained in a compact way assuming that the user has sufficient knowledge about that topic. But when this is not the case, the user can always retrieve additional information by clicking on the help button that provides the user with the extra information needed. This extra information is shown in the Theory-frame (see section 4.2). Most of the times this additional information consists of an extended version of the theory already explained.
4.2 Implementation

From all the requirements listed in Chapter 2 the following are of importance for the graphical user interface of the ELSET system:
- it should be easy to use
- it should contain a virtual piano
- it should contain a music stave
- it should contain an option for inserting notes in the music stave
- it should contain a frame where the exercise is explained
- it should contain a frame where extra information or help is shown
- it should contain a frame where feedback to the user is shown
- it should contain controllers for manipulating the music-stave, playing & stopping music-files, melodies, etc.

The implementation is these GUI requirements is explained in the rest of this section.

Virtual piano and music-stave
Like mentioned before, the most important features or tools in the program will be the virtual piano and the music-stave.
jMusic provides a nice interface for defining a music-stave. An advantage of this interface is that one has to specify only which note has to be written on the stave and jMusic will automatically write the correct note on the right position on the music stave.
The virtual piano and the music-stave must both be visible in the User Interface, so when for example the user plays a part on the piano the corresponding notes will be automatically visible on the music-stave.

The jMusic package contains thus its own music-stave accompanied with procedures for the usage/editing of it.
A music-stave instance looks like this:
As can be seen, the key signature and the time signature are shown first. This can be changed when necessary by setting the attributes of the music-stave as desired. The user has to click on the grey part of the music-stave to insert a note. When the note is inserted, that part of the stave becomes black and the grey part moves a little bit to the right so that the user can insert more notes. The music-stave can be as long as you want; every time a note is inserted, the stave is extended.

While jMusic offers an implemented music-stave, it does not provide an implementation of a virtual keyboard. A keyboard has to be designed and linked in some way to the music-stave of jMusic. Pressing a key on the keyboard should be visualized, the corresponding sound should be heard and the played note should be visualized on the music-stave.

The keys of the keyboard are modelled separately in order to combine them later on. On a keyboard five different keys can be identified:
- White key with black cutouts to the right
- White key with black cutouts to the left
- White key with black cutouts on both sides
- Black key

One by one every key can be associated with an appropriate virtual drawing in Java. After this description of the keys is accomplished, several instances of the keys could be combined in order to get the entire keyboard. By merging all the keys in the right order and on the right positions, the following is the result:

![Keyboard Image]

Playing the corresponding notes is then easy, because every key on the keyboard can be associated with a specific pitch-number. The sound connection with the soundcard of the computer is automatically done by the sound-device-class of Java. Once every key “knows” which note to play, the corresponding sound can be heard through the speakers of the computer.

Visualizing the played notes on the music-stave can then be done by means of jMusic. By looking at the pitch-number of the pressed key on the keyboard, a music-note can be inserted at the right position on the music-stave.

The GUI of ELSET allows the piano and the music-stave to be visible and accessible at all time during the exercises. Also, they are always on the same place, so that there cannot be any confusion. In cases where the buttons for inserting notes are needed instead of the piano, the piano is replaced with a set of buttons. The user won’t see any difference in the interface besides the replacement.
Outlook of exercise-windows

Taking the requirements listed in Chapter 2 into consideration, and in agreement with Ida Vujovic, the flow of the windows has been designed as follows:

Every exercise is implemented in its own window. So, if the entire ELSET application consists of 50 exercises, then there will be 50 separate windows for the exercises. From the requirements it also follows that each exercise-window should contain the following components:

1. A music-stave: the user must be able to see his input on the music-stave and to edit it.
2. Notes: an option to put notes on the stave manually should be available.
3. A keyboard: the user must be able to play on it (with the computer mouse).
4. A text area where the exercise is described.
5. A text area where some additional theory, help or tips can be given.
6. A text area where the answers to exercises are shown.

What must be noted is that in the screens where there is a keyboard, there is no need to put the option for inserting notes on the music-stave. The user will either use the keyboard or insert notes on the music-stave, he will not get exercises for which he will have to use both features simultaneously. So the screen either contain a keyboard or buttons for inserting notes.

Because the program is to be used at home and there is no teacher to whom questions can be posed, at least a clear way of navigation and usage of the program must be achieved. To do so, every exercise-window has the same structure: all buttons are always on the same place and they always have the same functions. Buttons which functionality is not required for the given exercises are disabled.

An exercise window looks like this:
(see figure on the following page)
The First Motif

1. Do you recognize this melody? It is a simplified version of Mozart's motif. Look at its notation and play it on your instrument.

2. Sing the motif by note names (written under notes).

3. Click on the buttons for different accompaniments (before the accompaniment starts, there are four "empty" beats, to set the tempo).

For exercises where the user does not need the piano, but buttons for placing notes on the music-stave the screen looks a little bit different. We replace the piano with the buttons for the different notes.

The First Variation (2)

1. Write down the motif you've just played (click below to hear it again).

2. Click on "play my answer" and listen. Did you write it correctly? Click on "evaluate" to get feedback.

3. Now look at the notation and sing the melody (on a neutral vowel). Also sing by note names. While singing, tap the beat (hit the table with your left hand on every beat).

To be completely sure about tapping the beat for this motif, click play to listen to the same melody with tapping in the background.
It can be seen that the screen is divided in several parts. On the right there are two areas: the upper one is for providing additional theory, help, or tips, and the lower one is for showing the correct answer providing the user with a feedback. Whenever the user wants to ask for help or for more information on a topic, this additional information is given in the special ‘Theory-area’. In cases where this information requires more space than provided in the relevant textbox, a new window is opened on top of the exercise-window while the former is still visible to the user. In this way the user can always easily find his way back to the exercise.

The upper left section is for describing the exercise; all the information needed to complete the exercise is given in this section. Multiple choice questions and questions that don’t require the keyboard are also given here.

The lower left part of the screen is for the music-stave, the piano or the note buttons. The music-stave is always visible and always on the same place. Below that there is either the keyboard or the set of buttons for inserting notes. The control buttons next to the piano and music-stave offer the possibility to control everything that has to do with sound, music-files, stave and piano features. Files can be played and stopped when needed, the notes on the stave can be played if desired, etc. In case that some buttons don’t need to be used, they are disabled. The buttons are only enabled when they are required for the exercise.

When answering the questions or doing the exercises the user will expect feedback. This feedback is always given in the same way in case of the same kind of questions.

- If a multiple choice needs to be answered, the feedback is given in the area next to the question (or options)
- When the user needs to do some exercises on the piano or on the music stave, feedback is shown in a separate window where the music-stave is shown, with additional feedback.
- In all other cases feedback is given in the ‘Answer-area’, in the bottom-right corner.

**Menus**

Exercises need to be done in sequential order to be a good learning-process, but for several reasons the user might be interested in just one type of exercises or wants to go to one particular exercise. This can easily be done by looking up the desired exercise(s) in the program menu. Besides the ‘Back’ and ‘Previous’ buttons in the exercise-windows, the program-menu is a perfect way for navigation.

The standard menu-bar of Java application window looks like this:

```
File  Edit  Help
```

This menu-bar can be edited in the Java-code, which makes it possible to insert any desired menu-items.

In agreement with Ida Vujovic the program has been made to contain exercises of different disciplines. The user must be able to choose from these. There should be a Table of Contents so that the user can go to any random exercise. Also an option to go directly to tests of the different lessons should be available. There is also a window with Tuning-exercises which the user must be able to access at any time.
So all these items have been included in the menu-bar, which now looks like this:

![Menu Bar Image]

For all the different menu-items a new window is made with the appropriate content. For example, when the user clicks on "Table of Contents", a new window will open containing a table of contents which looks as follows.

![Table of Contents Image]
4.3 Classes

The Exercise_frame is the main class of the ELSET program. Every exercise is an instance of this class. All exercise windows contain a music-stave and in most cases a piano. It can be said therefore that Exercise_frame almost do not exist without these two important classes Keyboard and ExtendedStave. These are linked to each other; the Keyboard is aware of the existence of the music stave. Exercise_frame is the one class that controls everything. Every operation is started here. Evaluating the user’s answer, playing files, giving extra information, etc., are all started in this class. This class contains instances of Keyboard and ExtendedStave and all the operations for the two are controlled by Exercise_frame.
The Keyboard class controls the virtual piano in the exercise window. For a detailed explanation of the functions, see section 4.2.

ExtendedStave is a very important class. Almost every operation has something to do with this class. If it is loading music files on the stave, playing the user’s input, and checking the user’s answer, all these operations are supported by ExtendedStave.

Most important methods supported by ExtendedStave:

<table>
<thead>
<tr>
<th></th>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>drawNote(key)</td>
<td>This method draws the corresponding note on the music-stave according to the key pressed on the virtual piano. When a MIDI-file is loaded the key of the note is read from the file. The note is automatically drawn behind the previously inserted note.</td>
</tr>
<tr>
<td>2</td>
<td>deleteLastNote()</td>
<td>Delete the note at the last position on the music stave.</td>
</tr>
<tr>
<td>3</td>
<td>getNote(position)</td>
<td>Returns the note of the current phrase at a certain position of the music stave.</td>
</tr>
<tr>
<td>4</td>
<td>getRhythm()</td>
<td>Returns the rhythm of the phrase on the music stave.</td>
</tr>
<tr>
<td>5</td>
<td>setRhythm(number)</td>
<td>Defines the rhythm of the phrase as the value of number.</td>
</tr>
<tr>
<td>6</td>
<td>getPhrase()</td>
<td>Returns the current phrase on the music stave.</td>
</tr>
<tr>
<td>7</td>
<td>setPhrase(phrase)</td>
<td>Inserts phrase on the music stave.</td>
</tr>
</tbody>
</table>

For checking the user’s answer ExtendedStave in turns accesses answerCheck. These are some important methods supported by this class:

<table>
<thead>
<tr>
<th></th>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>drawNote(Boolean eval)</td>
<td>This method paints note red or black. The Boolean determines if the note must be painted red or not. Default = black</td>
</tr>
<tr>
<td>2</td>
<td>evaluate(note)</td>
<td>Evaluates this note with the corresponding note of the answer. Returns false if not correct.</td>
</tr>
<tr>
<td>3</td>
<td>chooseImage(pitch)</td>
<td>Chooses the right note according to the pitch from a large collection of images.</td>
</tr>
</tbody>
</table>

This class contains a collection of images, representing all the different notes and rests that can be inserted on the music-stave. Each note of the user’s input is evaluated one by one. The evaluation returns true or false depending on the correctness of the note. After the evaluation, the right image (note) is chosen according to the pitch. The note is then painted red if the input was not correct and it is painted black otherwise. This whole phrase is then sent to ExtendedStave, which will put this phrase on a new music stave and show this to the user.

For the sound connection, every sound goes to the MidiSynth-class. This Java-class approaches and gives access to the computer’s soundcard or device. The playing of sound-files goes (depending on the format) through the playMidi or mp3Player class, where the stream is converted and finally passed through to MidiSynth. For example, the playMidi class offers the following methods: setTempo(tempo), readScore(file), setNumerator(number) and setDenominator(number).
These methods are existing methods from jMusic, but are adapted to suit the program requirements. By using these methods a MIDI-file is read, and converted into the correct format to send it to MidiSynth, which in turn makes a connection to the computer’s sound device.

RandomMidi is a class that contains set of music-files belonging to particular exercises needing these files. These exercises require selecting music-files randomly out of a greater set of files. When doing such an exercise, the randomMidi is notified which exercise is being done at the moment. It selects then the appropriate files to randomly select from. By means of Java’s random number-generator a file is selected and played.
4.4 Exercises of the ELSET system

The course is divided in Lessons, which in turn contain 4 Parts. Every part consists of a few exercises, one window per exercise. Each window (or frame) is given a name which looks like this: fLesson-Part-Exercise. For example: f2-3-6 is the window for the 6th exercise of Part 3 from Lesson 2.

Until now three lessons have been implemented, giving a total of 148 exercises. The exercises were provided by Ida Vojvavic, who also told exactly what was needed for each page. For example, which files need to be played, what the correct answers are, how the feedback to the user must be given, what the key-signature of the music stave must be etc.

Every lesson starts with an Introductionscreen, and from there one can continue with various exercises. The last part of each lesson consists of a number of test exercises. Navigation is easily done by clicking on the backward and forward arrows to go to the next or previous exercise. Also clicking on the options in the menu-bar above can do the navigation.

In this paragraph three exercises are explained extensively. Flow-charts will be shown in all cases. The three exercises are selected in such a way that they are representative for all other exercises.

Exercise 1
The exercise “Exercise 1” is to listen to some melody and write the motif down on the music stave by means of the note-buttons (see picture below).
Step 1: Listen to the piece of music
The user must click on the button “Play” to hear the melody. This melody can be stopped any time by clicking on the “Stop”-button.

Step 2: Write down the motif on the music stave
Using the note-buttons the user can insert the desired notes on the music stave. First he has to choose notation and then click on the grey part of the music stave. Mistakes can be undone by clicking on “Undo”. For example the music stave can look as follows:

![Music Stave Example]

Step 3: Evaluate the answer
After the user finishes his input, he has to check his answer, by clicking on “Evaluate”. The program will evaluate the answer and will give the following feedback in a separate window if the input is not correct:

![Feedback Example]

This Answer is: Not Correct

The red-marked notes indicate the notes that are incorrect, while the black ones are correct.

During insertion of the notes, the user is also able to let the program play his notes entered so far. He will simply has to click on “Play my answer” to achieve this. If he realy has no idea how to get to the right answer, he can click on “Answer” which will show the correct answer in a separate window.

On the next page two flow charts of the most important procedures in this exercise are shown.
### Playing the music file

- **Exercise-window**
- **playMidi**
- **MidiSynth**

- **Exercise-window**
- **ExtendedStave**
- **answerCheck**

- ***Extract data from file***
- ***Send midi-data to Java-synthesizer***
- ***Make connection with computer’s sound device***

### Evaluating the user’s input

- **Exercise-window**
- **ExtendedStave**
- **answerCheck**

- **Get user-input**
- **Retrieve correct answer and check input**
- **Make new stave with black/red notes**

- **Show evaluation result to the user**
Exercise 2
In this exercise, the user will have to recognize melodies and select which of the three rhythmic phrases belongs to that particular melody. Each click on “Play” will load a different MIDI-file.

Step 1: Load a random MIDI-file
The user has to click on ‘Play’ to load a music file. This action will select a random file from a collection belonging to this exercise. This file can be repeated multiple times and can be stopped at any time by using the control-buttons that are next to the stave.

Step 2: Recognize the rhythmic pattern
As the user hears the rhythm of the melody being played, he will have to recognize this pattern in one of the figures (a, b or c) shown in the exercise-window. Choosing one of the options will automatically trigger the evaluation and give the result in the green box.
This will for example look like this:

y steady pulse on bass drum. Concentrate on the melodic instrument and look on the picture. Which pattern (a, b or c) matches the rhythm that was played?

NOT Correct
A flow-chart of the randomly selecting and loading a MIDI-file is shown below.

Exercise 3

**MOZART'S MELODY PLUS STORY ABOUT THE REGISTER**

And finally, let’s reveal the original melody from Mozart’s Piano Sonata. Click “play” to hear it without accompaniment. The rhythm of the phrase is given below.

1. Write it in your notebook and check the answer.

Does the notation surprise you?

2. The original motif and its variations are recorded in different registers. Listen to each, determine the register (octave) by comparing it to the keyboard and play it yourself. You will hear whether or not it is correct.
The task of “Exercise 3” is to listen to a melody, to write the musical notation of it in a notebook, and to check the correct answer. So there are different tasks that can/must be performed here.

1) **Playing the MIDI-files.** Using the control buttons the user can play and stop the files with melodies as many times as needed.

2) **Checking the answer.** By clicking on the answer button next to the task, a new window will be opened and the answer will be shown on a music-stave. This window can be closed when finished and the user can go on with the rest of the exercise.

3) **Retrieve extra information.** For this task the user can get additional information on the theory of “registers”. There’s a button ‘read about register’ which when clicked, shows the additional information in the “Theory-frame” as shown below.

4) **Playing the melody on the piano.** The user can play some melody on the piano to determine the answer to the questions. By doing this he can compare his answer with the MIDI-files played.
Chapter 5: Program evaluation

After having implemented the first three lessons of the course, it became clear that most of the requirements stated before have been met. All the exercises working with the piano and the music-stave can be completed without difficulties.

The program has not been tested with real users yet. After a part was finished, Ida Vujovic was the one to evaluate the exercises made. For every exercise she checked if the program did what it was supposed to do (correct files, correct answer, etc). So if the part that was implemented was approved by Ida, I could go on with the next.

One unforeseen problem is that this version of jMusic does not support the 3/8 or 6/8 notation on the music-stave. For now, one can only get x/2, x/3 or x/4 notations.

It must be said that users are required to read the user manual before starting with the program. Otherwise it will not become clear what to do exactly. There is some textual guidance in the exercises but that may not always be enough. The program is now compiled and stored on CD-rom, and can be used after installing the required Java Run-time Environment, which can also be found on the CD.
Chapter 6: Conclusion & Future work

The goal of this project was to build a software package for the Royal Conservatory of The Hague which should be used by first-year students. The program should consist of two main features, namely a virtual piano and a music stave. With help of the jMusic package I was able to make a connection between those two and implement other less important features. The implementation of the exercises turned out to be very time-consuming. In my opinion we really underestimated that part.

All in all, I think a program was developed that will suit the needs of the Royal Conservatory. Other existing programs didn’t have all the features the Conservatory wanted them to have (see Chapter 1). By adding these features (e.g., playing MP3-files and making it a learn-by-doing course) in a new program, the Conservatory has now a good basis for further development of it.

The result is a program that:
- Enables the user to do the exercises entirely on their own
- Offers the opportunity to ‘learn-by-doing’
- Contains exercises which will increase in difficulty as the user goes on
- Can be distributed over the Internet or on CD-Rom

For further development it would be nice if a connection can be made between the ELSET program and an external keyboard which is plugged in the MIDI-port of the computer. In this way the user will be able to do the exercises in a more common way, that is playing on a real keyboard. The virtual keyboard is working correctly, but the fact that it should be clicked with the mouse is a little bit time consuming and, hence, unnatural.

The jMusic package I used is not the newest version. jMusic itself is a very new package and is still being developed and improved. So it may be possible that newer versions in the future can offer more and/or better features for this program. For example the musical notation of 3/8 is not yet supported in the version currently used for ELSET. This problem may be solved with a newer version.

A good usability study on this package would be necessary. It is an educational software package, so the users must be satisfied with what they get as learning exercises. And by this I don’t mean the content itself, but the system and especially the user-interface.

After having tested the program with two random users (colleague students), the following can be concluded:
- It is an attractive program, which can get the user’s attention.
- Some of the exercises are being experienced as “fun”.
- After 2 or 3 exercises the user is willing to continue with the rest.

To me this is a good sign, the program seems to be acceptable, even interesting, to the users. However, the users that did the test were computer science students. Hence, these promising usability test results, might not hold for music students.
Appendix A: Process Report

This project was originally intended to be completed in 8 months. For the University of Delft a total amount of 320 working hours were required. The project started in January 2003. The following abstract will give a summary of the process of developing the program described in this document. The whole process was guided by Ida Vujovic who provided all necessary musical and educational information.

First thing to do was to analyse existing programs and while the Royal Conservatory was using Practica Musica at that time, that program have been analysed. From there on further requirements for the new program were formed. After that, the options for implementing a virtual keyboard and a music stave had to be considered. By searching on the Internet, I discovered jMusic which is a complete music package supported by Java and offered a lot of possibilities of working with a music stave as it already contained one.

Now the only thing to do was to implement a virtual keyboard, this could be done relatively easy by virtual painting the keys and defining them in Java-code. Putting them together and defining the tones to be heard related to each pressed key, required some extra study. The Java source code had to be screened to find out how to play the different tones.

But the most important thing, connecting the music-stave and the virtual keyboard still had to be done. With each key pressed on the keyboard, the corresponding note had to be drawn on the music stave. Therefore jMusic had to be analysed thoroughly, because I had no idea how the notes were drawn by the package itself. With help of jMusic, the code was adapted to listen to the keyboard for some action, and when this action (click on one of the keys) occurred, a note was drawn on the right position (corresponding with the pitch).

This connection took a lot more time than expected. Actually 1 or 2 weeks were planned for this, but it turned out to be a whole month.

The next thing to be done, was to find how to play MIDI-files using the Java sound tools. This was one hell of a job, because it was unclear how the sound tools are assumed to access the computer’s sound-device. It took 2 weeks to figure this out. Once it was know how to implement this, is was easy to figure out how to play the notes that were inserted on the music-stave. The latter was also one important feature that needed to be implemented, because it would be used in many exercises.

All the previous lasted for 3 months when we could actually focus on the exercises to be implemented.

Ida Vojovic specified all the different features that would be used in the exercises:

- insert notes on the music stave
- besides MIDI-files also play MP3-files
- multiple choice questions
- play music files at random
- evaluating user input on the stave
- tapping rhythm

We decided to first figure out and implement these features before we started with the exercises.
To insert notes, the program should offer the possibility of choosing notes and place that on the stave. How to do this?

There are 12 different notes/rests that can be inserted, a specification was needed for every single note or rest. With help of Ida, every note or rest could be defined in Java, ready to be drawn on the stave by jMusic. But then the next problem came up: the jMusic stave has its own action handler, so clicking on the music stave by us had no influence. So there must be figured out, how to bypass this action-handler. Again it took longer then expected to figure out how to do this, and so another 2 weeks went by.

The next feature that was implemented is the evaluation of the user-input on the music stave. The big question here was how to visualise the result, that is how to distinguish the correct from the incorrect input.

After some brainstorming, we decided after evaluation the user input should be opened on a stave in another window on top of the main-window (as a pop-up) where the incorrect notes are painted red and the other are black. This wasn’t very difficult, but all the red notes had to be added to the set of available notes for the music-stave, so it was very time consuming.

Beside MIDI-files, also MP3-files are required to be played. After again consulting the Internet, we found an interface that provided a Java-layer that converts a MP3-audiostream into a standard WAV-audiostream that can be played by the normal Java-synthesizer. This was one of the easiest problems that had to be done.

Answering multiple choice questions is very straight forward and also this didn’t take too long. For the files being played at random a new class had to be written. This was not a problem, it was more some programming and testing.

The last feature needed was the tapping of the rhythm. Once it was figured out how to do this, several problems came up again. And the most important that was standing in the way was the timing of the tapping and the tempo. Together with Ida I spent some major time trying to improve this. Everytime there was something else that was not entirely correct. Eventually, after a month this feature reached the level we wanted it to be.

Now that all necessary was available, we were ready to start implementing the exercises. Every exercise was specified exactly by Ida: what text should there be, what answers are correct, how to visualise these, and so on.

After every part she would evaluate the implemented exercises, and do them as if she is the student. We spent hours sitting together changing some things after every evaluation by Ida.

Implementing those exercises also took much longer then expected. I spent almost 2 weeks on average for every part. Each lesson consisted of 4 parts, and so far 3 lessons are implemented.

Designing the menu’s was not the most difficult and time consuming task, this was done within 2 weeks.

What also took a lot of time was figuring out how to compile the java files into one single executable file. Until I discovered NativeJ, with this tool I was finally able to make an ‘.exe’-file.

What can be concluded is that some parts were very underestimated, parts that were not expected to be that difficult or time consuming, like implementing the exercises. Also a lot of unforeseen problems arose at every stage, some of which couldn’t be solved at once. Unavailability of both parties for some period and due to health problems of Ida the project lasted just too long for all of us.
Appendix B: The user’s manual for ELSET

1. Getting started
No specific installation of the program is required. You can start the program directly from the CD-rom.

But first you have to make sure you have a Java Runtime Environment installed on your computer. This JRE is required to run Java programs. The JRE needed is version 1.4.1 or higher.
On the CD-rom you can find an installation file, named “j2sdk-1_4_1_01-windows-i586”. By double clicking on this file, an install shield will be opened which will guide you through the installation of the JRE on your computer.

After the installation is done, click on the icon named “KC Application”. This will start the program. You’ll see a welcome window with some information, from where you can go on.

2. Closing the program
When you are finished doing your exercises you can simply close the program by clicking on the option “Go to…..” in the menu-bar above and choose for “Exit”. When you do this a small window will appear asking for your confirmation. Click “Yes” if you really want to exit or “No” if you changed your mind.

Very important! Don’t close the program by clicking on the exit-cross in the top-right corner of the window. This will only close the current window, it will not end the program.
3. The User-interface

The user-interface is kept very simple as you can see below. For every exercise the window structure is nearly the same. There is only one difference: some exercises require a piano (or keyboard) while others don’t. For those exercises that don’t require the piano, the piano is replaced with a set of buttons for placing notes on the music-stave.

**Exercise-frame:** In this part of the window the exercise is being described. In many cases it also contains a number of play and stop buttons for playing the midi-files belonging to that exercise. Sometimes the user is asked to answer some questions in the form of multiple choices. This is the main part of the window, where guidance and explanation are given.

**Theory-frame:** This is the part of the window where extra information, help and tips are being displayed. When the user for example clicks on the “help” button in the exercise-frame, the help-text appears in this window.

**Answer-frame:** The correct answers to the questions from the main frame are shown here. Like in the Theory-frame, the text will appear in the Answer-frame only on the request of the user (clicking on the related button). In cases where the answer is a large image it will appear in a new window.
SET OF BUTTONS FOR INSERTING NOTES ON THE MUSIC-STAVE

**Music-stave:** This music-stave is one of the most important features of the program. By placing notes on the stave, the user can create (and listen to) own melodies or transcriptions. In other exercises the musical notation of some MIDI-files will be shown here, and the user is able to edit it or adjust (change) the existing notation.

**Piano:** Another important feature of the program, the virtual keyboard. The user is able to play on this keyboard by clicking with his mouse on the keys. The corresponding note will appear on the music-stave.

**Functional buttons:** With these buttons the user is able to perform operations related to the piano and/or music-stave.

- **Undo:** removes the last note from the musical notation
- **Play:** plays a midi-file
- **Repeat:** repeats the last played melody
- **Stop:** stop the current file being played
- **Evaluate:** Evaluate the answer given on the music-stave or the performance of a rhythmic exercise
- **Answer:** Shows the correct answer
- **Start Answering:** Start the answering of the rhythmic exercises
- **Play my answer:** plays the notes that are currently on the music-stave.

In every exercise screen you will see a menu bar above with the text “Go to ...”. Clicking on this enables the user to choose from the different options, which are described in the next section.
4. Structure of the program

The course is structured in the following way:
It contains 3 Lessons, which in turn consists of 4 Parts. Every part has a number of exercises, for each exercise a different window. Every window is given a name; for example f2-3-6 means the 6\textsuperscript{th} exercise of the 3\textsuperscript{rd} part from Lesson 2. Every 4\textsuperscript{th} part of a lesson is a test belonging to the corresponding lesson.
The user can do the exercises in sequential order, or you can jump to a different exercise. This can be done by means of the menu-bar above. This is visible in every exercise screen. This menu contains the following options and choosing one will open a new window:
- Table of contents:
  In this window the user can choose to go to any part of any lesson he wants.
- Tuning:
  In this window the user can practice a number of tuning exercises.
- Disciplines:
  As all the existing exercises can be divided according to their discipline, the user can also choose the discipline he wants to practice and he will see all the corresponding exercises listed from which he can choose.
- Tests:
  As stated before every last part of a lesson is a test. Choosing for this option in the menu will give the user the possibility to go to the tests directly.

5. Navigation through the program

As soon as the application has been started, the user will see the “welcome-screen”. This screen has the same structure as the usual exercise screen, but has no further meaning. As noted before, the user can choose from the different options in the menu-bar to go to the exercises. Choosing one of these options opens the respective window from which he can proceed.

Once the user has chosen one of the exercises, he can not return to the “welcome-screen”. But he’ll always see the menu-bar with the options to choose from. Or he can just click on the “next” or “previous” button to go to the next or previous exercise.
6. Performing the exercises:

a) playing on the piano
The user can play on the piano by clicking on the desired key with his mouse. He will hear the corresponding sound of that key, and a note will appear on the corresponding place on the music-stave.
If he made a mistake just click on the button “Undo” to remove the last played note.
When the user finished playing he can click on the button “Play my answer” to hear his input.
He can stop the play backing of your input at any time by clicking on the “Stop”-button. To here the original MIDI-file (in case of a dictation) he can click on the buttons “Play” and “Repeat”.

b) placing notes on the stave
There are exercises where the user is asked to write down the melody on the stave. This can be done by means of the set of button which are visible in stead of the piano.
   1. First choose the desired note or rest from the button-set by clicking on it.
   2. Then click on the desired place in the gray area of the music-stave to insert the note or rest. This note/rest will be inserted after all the previous inserted notes/rests. It is not possible to insert a note/rest somewhere in between.
   3. If the note/rest appears on the wrong place, the user can adjust its position by moving it (and other notes) up or down with your mouse.
   4. In case he made a mistake he can click on “Undo” to remove the last inserted note.
Like with the piano, the user can also hear your input by clicking on “Play my answer”.

c) doing rhythmic exercises
In the rhythmic exercises the user is asked to tap a certain rhythm.
This can be done in the following way (read the exercise carefully for additional or more detailed instruction):
   1. Click on the button “Start answering” next to the music-stave. Nothing will happen yet.
   2. Tap once on the space-bar. This will start a beat in a certain tempo.
   3. The first tap that started the beat-sound is at the same time also the first note of your exercise. The user shouldn’t make pause before continuing to tap the required rhythm.
   4. When he’s done click on the button “Stop” to stop the beat. At the same time an answer-window will open with the evaluation of his tapping.