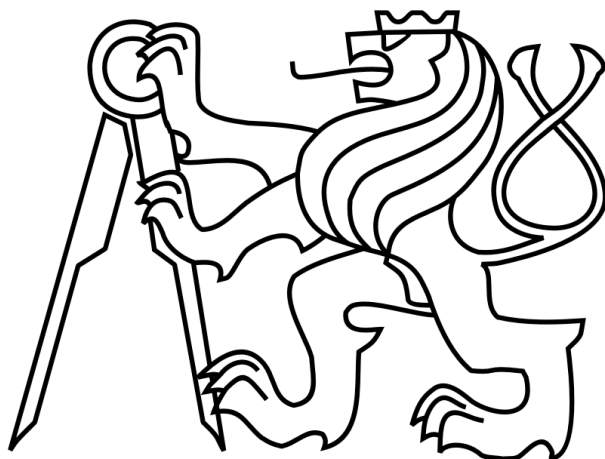


České vysoké učení technické v Praze

Fakulta elektrotechnická

Katedra počítačové grafiky a interakce



Bakalářská práce

Music Score Authoring Tool – Nástroj pro tvorbu notového zápisu

Edvard Rejthar

Vedoucí práce: Ing. Adam Sporka, Ph. D.

Studijní program: Softwarové technologie a management, bakalářský

Obor: Web a multimédia

Poděkování

Děkuji svému českému vedoucímu práce panu doktoru Sporkovi, že můj zvláštní českoanglický případ vzal na svá bedra, svému anglickému vedoucímu práce panu doktoru Saadu Aminovi, že se snažil porozumět hudbě, i když neznal ani notu C, a panu profesoru Žárovi, že mi ukázal Leninův spisek ve své knihovně.

Prohlášení

Prohlašuji, že jsem práci vypracoval samostatně a použil jsem pouze podklady uvedené v příloženém seznamu.

Nemám závažný důvod proti užití tohoto školního díla ve smyslu §60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne 24. 5. 2013

.....

Abstract

The easiest way of music notation is plaintext¹. However, plaintext cannot carry all the information needed (length of notes, rhythm), and the author cannot turn the plaintext into a real music score.

With a software that analyzes the plaintext and comprehends it as a melody, all that an author writes would be automatically recognized and exported to music score in the form of printable image or playing MIDI² file.

The aim of my project is to facilitate the process of music writing.

The application <http://mordent.cz> has been launched. It tries to combine the advantage of easy plaintext notation with the comfort of a real music score.

Mordent has been designed as an online freeware so that it can enjoy popularity.

Abstrakt

Nejlehčí způsob záznamu hudby je prostý text. Ovšem prostý text nemůže nést veškeré potřebné informace (délky not, rytmus) a autor nemůže prostý text jednoduše změnit v hudební partituru.

Se softwarem, který analyzuje prostý text a rozpozná v něm zaznamenanou melodii, se všechno, co autor napíše, automaticky převede do hudební partitury ve formě obrázku nebo přehratelného MIDI souboru.

Cíl mého projektu je zjednodušit proces hudební tvorby.

K tomu byla spuštěna aplikace <http://mordent.cz>, která se pokouší zkombinovat výhodu snadného záznamu not v prostém textu s komfortem opravdové partitury.

Mordent byl navržen jako online aplikace zdarma, aby si získal přízeň uživatelů.

¹ Term music plaintext is explained below: 1.5 Notation

² MIDI - Musical Instrument Digital Interface. See in the Definitions: MIDI, p 42

Contents

Chapter 1	Introduction	1
1.1	About this thesis	1
1.2	Aim: Mordent, music notation software	1
1.3	Thesis structure	1
1.4	Music theory	2
1.5	Notation	4
Chapter 2	Background	7
2.1	Other software on the market	7
2.2	What is different in Mordent	10
2.3	Comparison table	13
Chapter 3	Design	14
3.1	Theory	14
3.2	Required components of the application	16
3.3	Conventions	20
Chapter 4	Programming	21
4.1	Description: programmer experience	21
4.2	Application structure from the outside view	21
4.3	Login: LoginRadius	22
4.4	Performance boost	23
4.5	Source code	23
Chapter 5	Result	28
5.1	Description: user experience	28
5.2	First version conclusion	31
5.3	Further development	33
Chapter 6	Gained experience	36
6.1	Hard decisions the author made	36
6.2	Troubles encountered	36
6.3	Lessons learned	39
6.4	History log	39

Figures

Figure 1: Ordinary song.....	3
Figure 2: Example of paper-like sheet notation on computer (1).....	5
Figure 3: Note pitch and duration.....	5
Figure 4: Finale demonstration (4)	7
Figure 5: Sibelius demonstration.....	8
Figure 6: Studio Recording Session (8).....	9
Figure 7: Noteflight demo (8).....	10
Figure 8: Demonstration of a Czech traditional song.....	12
Figure 9: Used systems (brief).....	21
Figure 10: Used systems (extended).....	21
Figure 11: Login options in Mordent via LoginRadius.....	22
Figure 12: JavaScript request duration for editing	23
Figure 13: Homepage in Opera 11	28
Figure 14: Editing - welcome text	29
Figure 15: Editing - chord change	29
Figure 16: Editing - melody line.....	29
Figure 17: Shortcuts menu - special key pressed.....	30
Figure 18: Rhythm selection	30
Figure 19: Song details page in Chrome 15	30
Figure 20: Export - chord in marks	31
Figure 21: Export - chords in stave	31
Figure 22: Privacy settings	33
Figure 23: New editor functionality – menu buttons, highlighted chord in graphic staff	34
Figure 24: Main page button layout	34
Figure 25: "Decomposition chord" game	35
Figure 26: "Single tone" game.....	35
Figure 27: One key shortcuts in the current version.....	37
Figure 28: Strange behaviour of SQL command.....	38
Figure 29: Homepage in IE 8	39

Chapter 1 Introduction

1.1 About this thesis

This thesis has 11673 words. Formerly, it has been elaborated for Coventry University where the author had been dispatched from CTU. Since then, the application changed and many new features were added. Therefore, the thesis has been slightly modified so that it reflects new functions. Then, Further development subhead (p 33) contains the report from user testing that passed in December 2012 and list of changes that were implemented till May 2013.

If in electronic form, all **cross references** are clickable hypertext links. Numbers in parentheses point to references in the end of the document.

Personal or interesting remarks are in the box.

1.2 Aim: Mordent, music notation software

Mordent is online music notation software that parses plaintext into MIDI, PNG, and PDF files. Its aim is to allow users to quickly create music score from basic musical arrangements for e.g. piano and guitar written in plaintext.

The user just inputs some plaintext, such as the following. No installation is needed:

```
C7      G      C      F
London's burning, London's burning.
C      F9      C      F
Fetch the engines, fetch the engines
```

Since it is plaintext only, it inherits the attributes of music editing in Windows Notepad. The user can search and replace in the text, paste it to the body of an e-mail... they have full control over his score.

Mordent recognizes lyrics, melody and chords.

1.2.1 Objectives

The objectives have been set in the Proposal (further see 5.2.2 Objectives, p. 32):

Provide a domain and a hosting for the application, place video tutorial on YouTube, process different notation manners, and implement versioning, importing, pdf export, social login, and collaborative possibility.

1.3 Thesis structure

The thesis will lead the reader through the whole process of the development of Mordent, from the origins of the idea to the successful accomplishment and the lessons learnt.

First, the reader will be introduced to the indispensable music theory so that they can understand different approaches to music notation.

The second chapter focuses on how Mordent differs from other applications. A brief evaluation of other products currently on the market will lead to the explanation.

Subsequently, the Mordent design will be presented. We will describe the outlining process of Mordent, and the initial questions and challenges.

In the fourth chapter, the real application will be described in detail, both from the perspective of the server (supply side), and of the client (demand side).

In the chapter on results, the user experience will be tested.

Finally, in the last chapter "Gained experience" (p 36), you will find author's learning outcomes and troubles encountered during the Mordent's development.

1.4 Music theory

Some basics of the music theory need to be highlighted in order to enable all readers to understand what does Mordent do and why. This chapter will help the reader to truly estimate the added value of the Mordent.

Our definition is neither exhausting, nor complete, since it serves only the purpose of orientation in the Mordent application.

1.4.1 Note

The note is the basic unit of a song. It is a tone sung which has certain pitch and duration. Pitch indicates the exact frequency³. Duration is a time segment, which length depends on the song tempo. A sequence of notes forms **melody**.

In a simplified way, the pitches are referred as letters⁴:

C D E F G A B

The syllables sung in the background of a melody form lyrics.

1.4.2 Chord

Some music instruments are able to play multiple notes simultaneously. In order to shorten the notation, several notes that are played together are referred by the letter of the principal note followed by other letters or numbers, widely known by musicians. The example of such chord marks can be as follows:

C D7 Em Fsus4 G/B Amaj7 B7+

Some basic chords are used in the majority of songs, others can be invented to create a unique colour of the song. Others indicate information which can be interpreted in a different way on piano or on guitar.

Chord *G/B* indicates the bass line, differently interpreted in piano and guitar playing.
Chord *C5+/9* can be easily played on piano, but it is so complex that we would play just *Cadd9* on guitar.

The first letter is referred as the root of the chord, or the principal note the chord is built upon.

³ Let's assume now the pitch indicates the exact frequency. Let's omit some facts that are not important for Mordent: „A = 440 Hz“ international pitch standard; „C#“ = „Db“ problem; letter repeats every octave

⁴ English scale in C major

1.4.3 Stave

A stave is a set of five horizontal lines and four spaces where notes are placed. We will use chord stave, melody stave, and lyrics stave.

In polyphony, multiple voices can be written down in one stave or every voice can have its own stave. For piano playing or choir singing, there is often first stave for soprano and alto voice and the other for tenor and bass voice. For orchestral playing, every musical instrument has its own stave.

The system of staves forms music **score**.

1.4.4 Music score

Music score is the form in which songs are notated. Musicians and singers are able to perform a piece of music from the score.

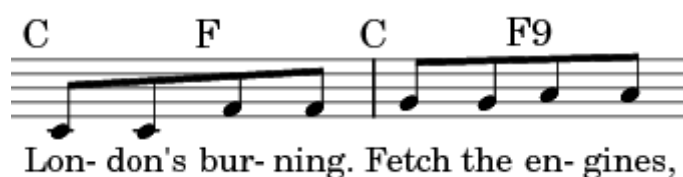


Figure 1: Ordinary song

The Figure 1: Ordinary song shows how a score can look. The five lines in the middle form the *melody stave*.

Below the five lines, there is a text or the *lyrics stave*. It is important to understand that each syllable corresponds to a note. Finally, there is the *chord stave* above. Even though the chords notes could be written in the melody stave, practical experience led the musicians to write the chords above the melody stave in the form of chord marks.

1.4.5 Song form

Ordinary songs often consist of several verses and refrains.

Compared to classical music compositions where multiple and complex staves are used, forming unequal musical parts such as *adagio*, *largo*, *vivace*, *allegro*, or *andante*, the musical score of ordinary songs can be relatively short.

For example, lyrics "*Yesterday*" from The Beatles consist of four verses and two refrains. It means it is sufficient to write the melody and chords for the first verse and the first refrain only. The other verses and refrains follow the same melody and chords, accompanied by different lyrics.

1: Yesterday all my troubles seemed so far away ...
2: Suddenly I'm not half the man I used to be ...
R: Why she had to go I don't know ...
3: Yesterday love was such an easy game to play ...
R: Why she had to go I don't know ...
4: Yesterday love was such an easy game to play ...

1.5 Notation

1.5.1 Digitalization

Both in commercial and non-commercial sector, a high number of music pieces are created. Music digitalization brings the author many benefits:

- sorting – keeping track of their songs
- storing – their songs can be backed up easily
- sharing – make themselves known in public
- printing – creating score whenever needed
- playing – listening to their music

The most evident disadvantage of music digitalization is the time consumption. We have to manually rewrite the score into the computer if we do not have an OCR software, capable of handwritten music recognition⁵.

1.5.2 Notation in general

The art of music notation dates deeply before the digital era. During centuries, many marks and customs have been invented. However, what is very convenient to be drawn on paper, might be on the other hand very hard to be written with the help of ASCII characters the keyboard limits us to.

1.5.3 Paper-like sheet notation on the computer

Thus, the computerized music notation becomes painful. Multitude of software developers took the challenge of sorting it out.

In general, they try to simulate the paper notation as much as possible.

User interface resembles the paper (see 0): The staves can be seen, the clefs and all the marks that would have been drawn on the paper. Output can take the form of a MIDI file or a printed sheet.

⁵ The author did not find any reliable free software with handwritten music recognition as it is not an ordinary functionality.



Figure 2: Example of paper-like sheet notation on computer (1)

1.5.4 Plaintext: Natural notation on the computer

Typical user

Let's suppose that we have an average computer user. He is gifted in music, nevertheless he is not experienced in current music software.

Provided he wants to write some music score, it is doubtful that he start seeking the best music score software. It would take him too much time to install one and to learn how to use it effectively.

It is far more likely that he opens a text processor and writes notes and lyrics in plaintext. But without special musical marks! He would not use them because advanced Unicode signs are not easily accessible via the keyboard.

Rhythm

It has been said that every note has a pitch and duration. Take a look on the first note in Figure 3: Note pitch and duration". What can see a musician?

- The pitch C: it is below stave lines, on a separate line
- Duration 2 beats: its white and has a leg (half-note). A song with the tempo of 120 beats per minutes would contain 60 half-notes.



Figure 3: Note pitch and duration

How it would be written in plaintext? The musician would write pitches above the syllables, to make it as similar as possible to the paper-like notation. But without rhythm.

```
| c    d  
| Some text  
| g    a    b  
| And next line
```

There is no easy way to indicate the rhythm in plaintext. If written by the number, (2 for half-note, 4 for quarter note etc.) it would become impossible to write notes longer than the whole note.

```
| c2    d4  
| Some text
```

If written by fraction, we would lose the biggest advantage of plaintext notation – the transparency. It would look messy:

```
| c1/2 d1/4  
| Some text
```

Thus, the rhythm is often omitted in plaintext notation.

Examples of the use on the Internet

The *plaintext of chords and lyrics* is frequently used notation amongst songbooks on the Internet. Hereafter are two examples, demonstrating the widespread use of the plaintext notation:

<http://www.velkyzpevník.cz/zpevník/beatles/yellow-submarine> (English)

<http://www.guitaretab.com/a/astrid/311542.html> (Indonesian)

1.5.5 Natural vs. sheet notation

Had our project been of social science nature, we would ask volunteers to participate in a survey in order to support the following statements. But since this paragraph serves merely as an introduction, permit the author to rely only on his personal experience.

- It is easier to write in plaintext than to read it.
- It is easier to read the music sheet than to write it.

In other words:

People prefer to read the rich sheet notation while playing the music.

It is easier to buy food from local store than to cook. However, we would prefer home-cooked meal.
--

Chapter 2 Background

To understand the added value of Mordent, we will walk through the similar products on today's market.

2.1 Other software on the market

2.1.1 Finale

According to their webpage (2), Finale Music offer wide range of music programs, based on their main product, Finale 2012. The distribution comes in multiple price options⁶:

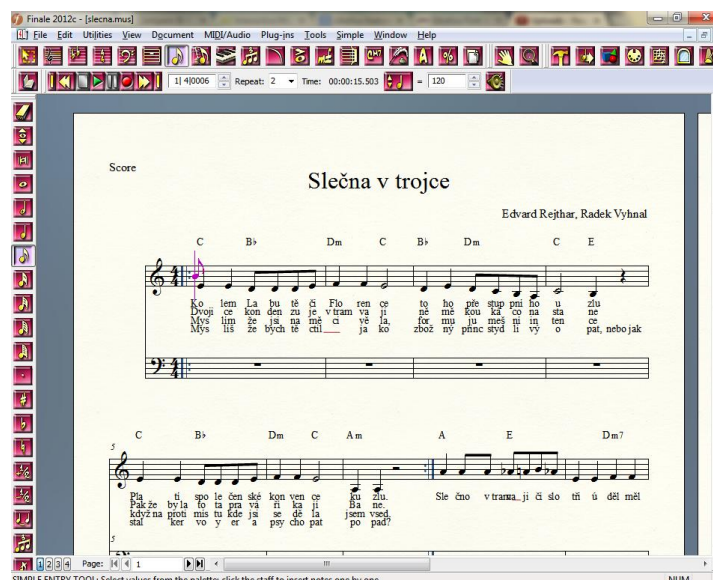
- Finale 2012 – up to \$600
- Finale PrintMusic – up to \$200
- Finale SongWrite – up to \$50
- Finale NotePad – free

Finale 2012 is full version, whilst other versions have reduced functionality. Then, SmartMusic (3) is offered for up to \$140 p/y. But it is software for practicing and not a music editor so we will not describe it further.

Take a look at the Figure 4. It is a snapshot of current Finale 2012 GUI. In the middle, a music score with two staves is written. On the left side, buttons for inserting notes are placed. On the right, there is a lot of buttons, functionality of which is not obvious. It took the author a lot of time to look up all shortcuts he had to use before some notation could be produced. If he had not used shortcuts, he would have clicked all the notes manually by the mouse which would have been very painful and lengthy.

There is no doubt, Finale is a great music editor. However for new user, Finale is a robust and too complicated product because the investment of few hours is required to master the input and editing methods.

Figure 4: Finale demonstration (4)



2.1.2 Sibelius

Sibelius is a direct competitor of Finale. It is seen well from the fact Finale advertise a notable discount if any client abandons Sibelius in its favor (5). It comes in full Sibelius 7 version and a limited Sibelius First version. Sibelius 7 Academic cost around \$180 for a license (6).

⁶ Finale products differences: <http://www.finalemusic.com/products/compare-finale-products/>

The developers implemented the ribbon as known from Microsoft Office products. It really enhances the first contact with the program: The control panel is situated in the upper part where all control elements are accessible via thematic tabs and subcategories.

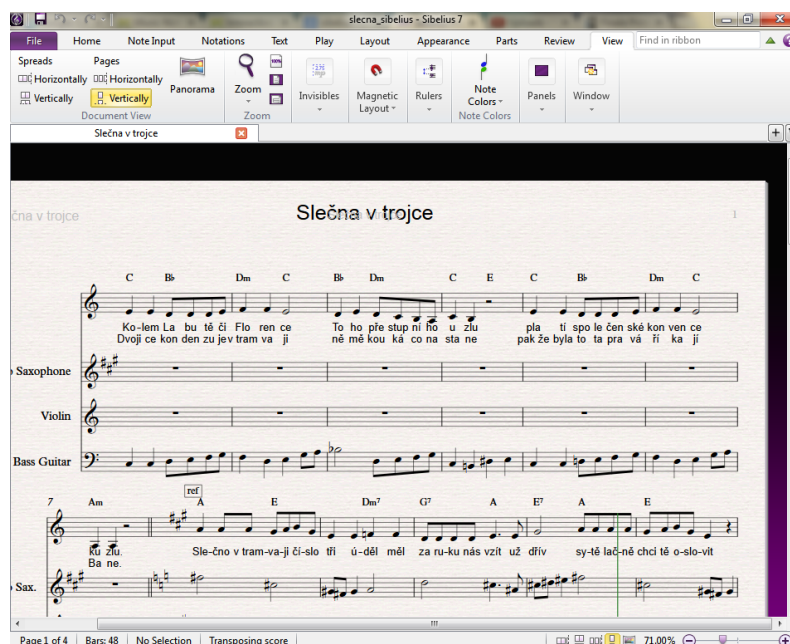


Figure 5: Sibelius demonstration

2.1.3 NoteWorthy Composer

Licensed version of NoteWorthy Composer costs \$50 (7) and in comparison with Finale and Sibelius, it is a tiny application of circa 30 MB only.

There is a common error of many applications to overrate the mouse over the keyboard. It is true that with the mouse pointer, user cannot get lost on the screen. But when they want to use an application on a daily basis, the use of shortcuts is much more convenient. If there is no possibility of using the keyboard, working with the application becomes exhausting.

NoteWorthy Composer has an inversed problem. The user interface is keyboard oriented, which would be an advantage if the same functionality could be acquired by mouse.

The new user is drowned in shortcuts and considers the application interface weird.

2.1.4 Studio Recording Session

Studio Recording Session is an old piece of software that has no official sources to be referred: unfortunately, no web page seems to be working properly of its developer, MidiSoft Corporation. Even though there was minimal keyboard support, the simple interface allowed creating music quite comfortably. Still, the application kept falling down in longer projects, and when the user needed to copy large blocks of bars, they often got lost.

Around the year 1995, it was an excellent tool for creating music, now is too old to be used.

In red circles on the Figure 6: Studio Recording Session, you can see the important parts of the working interface as marked by a program fan (8).

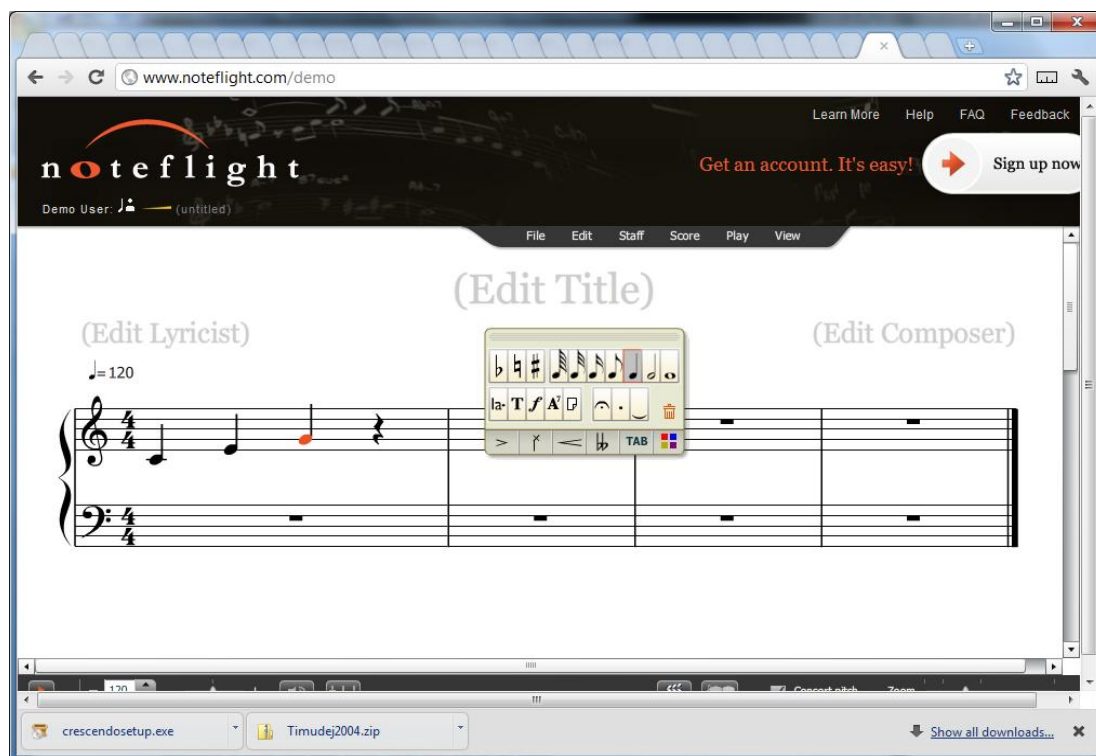


Figure 7: Noteflight demo (8)

2.1.6 Lilypond

Lilypond is a free software under GNU license (10). There is no visual part, no GUI. User just writes the plaintext in Lilypond syntax and then uses the software to generate the output (e.g. MIDI).

However, the syntax might be difficult to understand, and since Lilypond does not provide easily accessible error log and simply fails with brief error message, it might be hard to debug the plaintext.

Mordent uses Lilypond as an/the **output processor**. The PDF, PNG, and MIDI files are generated through Lilypond.

There are many external GUI for Lilypond, one of them is Denemo (see below).

2.1.7 Denemo

Denemo is a free software under GPL license (11), its purpose is to provide the GUI for Lilypond.

At first sight, everything looks like a perfect application. There is neither error in mouse nor keyboard interface, moreover, user can input notes via USB MIDI piano or just by singing in (microphone based analyzer is part of the program).

Nonetheless, the program keeps falling. It appears to be still under development and it would be wise to try the product again in few years.

2.2 What is different in Mordent

The other programs are mostly WYSIWYG⁷ as stated in 1.5.3 Paper-like sheet notation on the computer. Let us explain.

⁷ “What you see is what you get“ editor. See: WYSIWIG, p 42

2.2.1 Difference - advantages

Mordent tends to bridge the gap between plaintext and sheet notation, and tries to make easier both music writing and reading.

User experience:

As we saw in the previous chapter (Other software on the market), the plaintext notation is often neglected. The WYSIWYG way is prioritized. However the author considers it unhandy (as explained in 1.5.5 Natural vs. sheet notation).

The way Mordent converts plaintext into MIDI or sheet score, may motivate the user not only to create new songs but also to bring to life his old songs; the old songs which would be forgotten due to the laboriousness of converting them with current software.

What would normally take hours, it can be done in few minutes since the **plaintext is native (12) format** for Mordent.

Furthermore, **highly customizable chord marks are converted into MIDI** notes as well (see 3.2.2 Handy chords).

System configuration:

Generally, other software

- has to be installed
- & has to be paid.

A musician may not be skilled in system administration, thus he could experience troubles before running the software properly.

Mordent is **free** for use and accessible **online**. No special requirements are present. (Neither Adobe® Flash® Player in the browser is needed.) Moreover, **the registration is not necessary**: trying Mordent is as difficult as a single click.

2.2.2 Difference - disadvantages

Technology limitation

Since Mordent serves as a quick helper with simple songs (see 1.4.5 Song form), it is not intended to be used for tracks longer than several tens of bars. There is no possibility to notate complex music marks, polyphony, or multi-instrumental orchestral parts.

Paradoxally, there is no way in Mordent to notate the music mark "Mordent" (13).

Its functionality depends on the JavaScript framework jQuery so that non-standard browsers or devices without JavaScript do not have access to the editing functionality.

There is no official definition of what does “non-standard” browser mean. But we consider as “non-standard” every browser where some functionality despite being widely used by programmers’ world is not supported⁸.

Chord marks

There is a problem with chord marking. As seen in the Figure 8: Demonstration of a Czech traditional song, first chord is not above the first note but above clef.

In a similar way, second chord mark is not right above its note but hangs above the bar line.

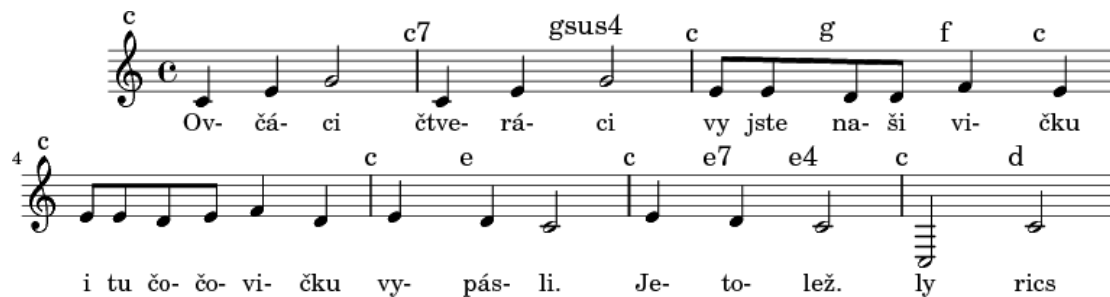


Figure 8: Demonstration of a Czech traditional song

This problem is further explained in 6.1.1 Lilypond.

⁸ E.g. the author will not implement twice the functionality of rounded corners. When Opera, Firefox, Safari, and Chrome recognize the CSS3 command „border-radius“ which is going to become standard, he will not spend an hour by drawing images for Internet Explorer 7.

2.3 Comparison table

We compare the discussed products in following categories: plaintext parsing ability, number of musical staves allowed, tablature availability, parsing of chord marks capacity, running environment and condition if the program is free or not.

Mordent and Lilypond are the only application that parse plaintext. As explained above, the plaintext parsing is the main purpose of Mordent – plaintext syntax in Mordent is much easier than in Lilypond.

On the other hand, Mordent can bear only two music staves – one for chords and one for melody. The other application support any number of staves, limited by product version in the case of Finale and Sibelius or by memory in the case of other applications.

The third column surveys the ability of guitar tablature export that is not available in current version in Mordent but is planned (see Future steps, p 35).

The majority of programs listed is able to notate chord marks. Neither NoteWorthy Composer nor Studio Session can mark the chords above the melody staff, the chords can be written only by its pitches directly in the staff – but the other programs can. However, Mordent allows user to define its own chord mark as explained in Handy chords (p 17).

Mordent and Noteflight run in browser, the rest runs as a stand-alone system application or in the command line without GUI (Lilypond).

	Plaintext	Staves	Tab	Chord marks	Environment	Free
Mordent	x	2	-	x	browser	x
Finale	-	8+	x	x	system	x ⁹
Sibelius	-	16+	x	x	system	-
NoteWorthy	-	30	-	-	system	-
Studio Session	-	50	-	-	system	n/a
Noteflight	-	n/a	-	x	browser	x
Lilypond	x	+	x	x	cmd line	x
Denemo	-	+	x	x	system	x

⁹ Reduced version Finale NotePad is free, the other versions are paid (see p 7).

Chapter 3 Design

The chapter describes the steps we took before writing the source code of the application.

3.1 Theory

First, we will take a look on the methodology, then to the task order, the way the ideas and the tasks were organised.

3.1.1 Methodology

We will apply the methodology of software engineering to create a web service.

Software engineering means “establishment and use of sound engineering principles to obtain economically software that is reliable and works on real machines efficiently.” (14)

Architecture

We will create a web based application of type client-server. Client (a web browser) will access the server over HTTP protocol.

High-level programming languages PHP (server side) and JavaScript (client side) will be used.

Application logic will assume calculations and data operations & storage.

Business logic will cover the functionality of the user interface (what user wants to do).

User will input the data to be processed and stored. When requested, an output will be provided. In other words, the user will not be bounded by a fixed sequence of steps from the beginning to the end.

To achieve that, we will use the OOP¹⁰. OOP is a methodology of software writing, characterized by conceptions such as: encapsulation, polymorphism, or object inheritance (15). Objects are elemental data structures of the modeled reality.

The functions will be independent of one another; they will not have to be launched in given order.

For example, song editing and save function will be independent.
Save is not conditioned by finishing of editing.

3.1.2 Milestones

- a) **Drawings.** Firstly, we drew notes and models on sheaf of papers till we were sure to have all the requirements in the head.
- b) **Core programming.** Then we programmed the core part of the application and installed the output processor.
- c) **Bugs.** Finally, we started to implement all tasks and bugs we had encountered.

3.1.3 Data storage

Idea storage

Since this paragraph consists of personal notes of the author, may the reader forgive its informal tone. The moment an idea appears, the author writes it somewhere down. No matter if he is at the lecture, in the bus, or

¹⁰ Object Oriented Programming

in the bathroom. Otherwise, he would come to the computer with a stressing feeling that he had forgotten something important. He cannot keep many things in his head at once.

- a) He do lists and drawing on papers.
- b) The information is written to the notepad as soon as he is sitting at the computer.
- c) When notepad seems messy, I drain the information into structured page on my personal MediaWiki installation.

The same technique the author used for writing of this thesis. So when he eventually started writing, the work had already consisted of ten pages, filled up with the structured chapter names and to-be-done notes.

Every night, he wrote how many hours he had worked and what he had done (see 6.4 History log).

Source code storage

As a working platform, we used integrated development environment NetBeans. The source code has been uploaded after every change we made so that the code was backed up in two places in every moment.

Besides, we did a complete zip of dated source files sometimes (not database) and this backup was stored in a separated place. (In the backup folders on an FTP, external disk or private profile on school computer.)

Since december 2012, we were using Subversion tracking system on Assembla¹¹.

3.1.4 Tasks and bugs categories

The task means a unit of work to be done. We will now describe the task classification and we will list some examples for every task type.

Current tasks

- **Major.** Mordent principal functionality. Must be done before we can proceed.
 - “liquid design – textarea resizes with the page + align with header ”**
 - “chord parsing”**
- **Minor.** Tasks that have to be done before application launches to public; however, the functionality is not crucial.
 - “multiple users”**
 - “shortcuts”**
 - “sent hint according to the line; lyrics hint, chords hint, note hint”**
- **Improvement.** Possible enhancement.
 - “chord parsing recognizes ‘sus’ mark”**
 - “staves can be added or removed on the fly, not only in the beginning”**

¹¹ <http://www.assembla.com>

- **Bugs.** Every time we get across some malfunction, we log it. Sometimes, it is not possible to repair it directly, for the concentration would be diverted.
- **Conditioned.** Minor tasks, for which we are not sure they would be useful. Time shows the manner the program is used. In the meantime, they can be developed.

"case sensitive major / minor (D major / d minor)"

- **Bureaucratic.** Nothing to do with the functionality. Has been promised to somebody.

"write the thesis"

"pay the hosting"

- **Future.** Interesting features that cannot be implemented right now. The feedback plugin was implemented recently so that users can tell the author the most desired feature.

"If I get many French users, I let them to change the shortcut letter 'q' because 'q' is in many French words."

- **Security.** Possible security holes to be tested.

Past tasks

- **Done.** What has been done.
- **Deprecated.** The functionality has been rejected.
- **Maybe done.** The author cannot recall if the task has been implemented. He could have done it quickly within the time range of another task.
- **Unknown.** The author cannot recall what the note was about. However, it is not wise to delete it, it may be useful in the future.

Other notes

- **References.** Useful articles the author may use in the thesis.
- **Log.** How long did the author work on which task.

"date – hours – task content"

- **Rubbish.** The author's personal category that he use everywhere. Rubbish is trash information which will not be ever useful and do not have to be sorted in any other category. Nevertheless, exceptionally the author is glad to have the possibility of seek something out from the past.
- **Triple 'x'.** Wherever is triple 'x' it means something has to be filled. Before anything is submitted, the author look for that mark in the text. (In thesis, in source code...) In addition, the author can specify the category of the mark, it boosts his orientation when fulfilling the tasks.

"xxx:enhancement"

"xxx:conditioned here comes 'case sensitive task'"

NetBeans IDE interprets triple 'x' as task. So that the author do not need to search for the string, he just click on the "display tasks" menu.

3.2 Required components of the application

Since Mordent will be online, it is clear that we will do a client-server based application; it will be written in PHP and JavaScript, since the author is skilled in that scripting languages and dispose of PHP hosting.

3.2.1 Core

The crucial part of the application is the editing window. User inputs inside plaintext notation which is transferred into database.

JavaScript has to interpret the user's input as staves, it will recognize lines as chords, melody, or lyrics; its separated parts will be recognized as chords, notes, and syllables.

Shortcuts

- must be **at disposition**, in order to allow user to work quickly
 - user will not use shortcuts if they have to seek it **deep in the FAQ**
 - **in every moment**, the user has to know what shortcut can be used
- Therefore, all possible shortcuts will be listed in the right panel.

3.2.2 Handy chords

The rules for chords are rather complicated. Many rules exist, several differences might occur depending on the country, few habits emerged¹². Amongst amateurish musician some chord notation rumors circulate, people like to create new chords... Neither the author, he is not an ultimate authority.

Mordent is not meant for professional use; it just converts the plaintexts according to the user wish and does not impose rules.

For example, the author is used to notate G^{\wedge} for *1st barré G* on the guitar, $G^{\wedge\wedge}$ for *2nd barré* etc.

If the user try to use non-standard chord in other software, it cannot be

- printed – non-standard notation is not permitted, or
- played – program do not recognizes the notes of the chord.

Chord type-hinting

Every chord mark written by the user will be analyzed and tried to put into notes according to:

a) Previous experience

What chords did the user use before? If he set notes for ***Dmaj7+/11***, the notes for ***Cmaj7+/11*** will be automatically counted.

b) Algorithm

The chord will be parsed for all known notation caprices.

c) Previous experience of other users

What chords other user used before? The entire database will be searched.

If not solved, the chord will be treated as basic major chord¹³ and user will be prompted to set the pitches manually.

¹² E.g. instead of flat sign \flat we write just small letter 'b'

¹³ We mean *C E G* for every chord beginning with *C*. (Or *C# E# G#* for every chord beginning with *C#*.)

3.2.3 Bridge between database and output processor

Export

The export is the very next aim of the application. The user has to obtain his input in a well formatted way.

It would be convenient to export following formats:

- PNG – user shares
- PDF – user prints out the score for the rehearsal
- MIDI – playable format which can be imported into any other application.
User would quickly convert his plaintext to MIDI and then edit the song meticulously in Finale or Sibelius (in order to finish details).

Possibilities

We could implement MIDI, PNG, or PDF output, but not all at once, since their specifications are complex. If there is already a free library that could serve as output processor, we will consider the implementation of MIDI, PNG, and PDF as time wasting.

As told above, finally we chose Lilypond as output processor.

Rhythm

As explained above (see 1.5.4 Plaintext: Natural notation on the computer / Rhythm), it is a great challenge to sort out how the rhythm should be marked in Mordent, a plaintext based application.

We would go as far as to leave the rhythm notation out altogether. However, **it would have made the export impossible.**

There is **no way to predict** the note duration with sufficient accuracy. For the note duration does not depend neither on the string length of its lyrics syllable nor on the line length.

Regardless, Mordent can predict the desired note duration based on the previous experience. If a line has five notes, it is probable that another line with five notes in would have the same rhythm. Mordent will propose that rhythm. Though it must be the user who decides.

Then, Mordent stores somehow the rhythm information and delivers it to the output processor when asked.

Lyrics and melody

It has been told that an element of melody is a note, an element of lyrics is a syllable (or a short word). The lyrics and melody are connected.

- every syllable has to be linked to its own note
- every note has 0..n syllables

User inputs:

c	c	f	f
London's	burning		

Output to the output processor (see Figure 1: Ordinary song):

	c	c	f	f
	Lon-	don's	bur-	ning

The above example shows that inner algorithm had to **link** the syllables to its notes. And recognize the ends of words in order to **attach hyphens**.

Chords

The user can whenever decide to export the tune with chord marks or with chords in different stave. So that Mordent must be able to deliver the chords both in the text representation and by their composing notes.

Rather than understanding what do the terms "chord marks" and "chords in stave" mean, take a look at the images exported from Mordent, such as Figure 21: Export - chords in stave, p 31

3.2.4 Song details

The layout of the page will be designed so that the user can transparently

- a) Set details of the song
Author, description, year of publication
- b) Set the privacy
Can others view the song? Edit?
- c) Obtain the sharing link
- d) Export
(Described in 3.2.3 Bridge between database and output processor / Export)

3.2.5 Role of the server

Server will keep track of users, songs, and chords.

3.2.6 Login

Let's assume users do not like to register unless they have confidence in the application and are convinced of its indispensability.

They don't like to:

- Share their personal information
- Being troubled with password safe storage
- Being asked every time for password

Me, personally, I hate logging.
If any application prevents me from proceeding further unless given my personal (or fictive) details, I leave.

So we put the condition that Mordent is accessible without registration or login. This makes the application design more difficult. Database will have to store the songs for temporary users, so the data might get scattered.

Since Facebook users have often their cookie in the browser memory, I might offer Facebook login. And in order not to prioritize Facebook over other social networks, it would be great to allow users to log in via Google+ or OpenID.

3.2.7 Additional user interface

Homepage has to be nicely designed in order not to discourage new users. Even epoch-making invention may be overlooked if not in a bearable design.

It will contain:

- brief information about the project
- direct access to the core functionality (editing)
- login
- list of songs of logged current user
- list of public songs of other users

3.3 Conventions

3.3.1 Naming conventions

- It is desirable to produce long, self-explaining method names.
- Comments formatted as PHPDoc, JSDoc
- We prefer to tag the variable type. We write rather *entityA* (array of entities) than simple *entities*. For an instance of song object, we name the variable *songO* rather than *song*.

Some examples:

- variable, method – **camelCase**
- class name – **UpperCamelCase**
- URL – **hyphened-text**
- file – **underscored_text**
- constant – **BIG_LETTERS**

3.3.2 Comments

- We write comments in the Czech language without diacritics.
Why not in English?
Since English is not my mother tongue, it might be incomprehensible. The author is the only coder, and the comments are written for him.
- Deprecated or obsolete code blocks are commented with prefix of the letter “x”. We may add reason for making the code obsolete as in the following:

```
| //Xhe's got id from the beginning: $_SESSION["id"] =  
| User::$himself->getId();
```

Chapter 4 Programming

The source code creation will be depicted.

4.1 Description: programmer experience

4.1.1 Overview

The length of source code we wrote for Mordent is about **170 k keystrokes**. There is 79 k of JavaScript files and 73 k of PHP scripts.

Besides, I use my own framework and database layer Sarotherodon (see 4.5.2 PHP framework Sarotherodon 4.5.2, p 24), its length is **111 k keystrokes** in current version.

4.1.2 Used systems

Basically:

- Server side scripting: **PHP**/Apache.
- Database: **MySQL**
- Processor output: **Lilypond**
- Server side scripting: **JavaScript**
- Markup language: **HTML**
- Styles: **Cascading style sheets**

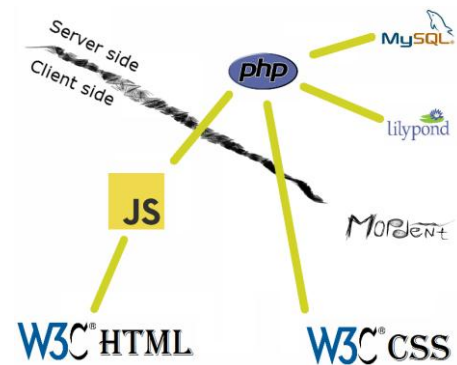


Figure 9: Used systems (brief)

On the Figure 10: Used systems (extended) you see several more details:

- PHP is connected to database via my own database layer.
- Both document object model and its styles are treated by jQuery framework.
- Lilypond generates files into cache. PHP provides these cache to the user.

Further details will be treated in the paragraph "Source code" below.

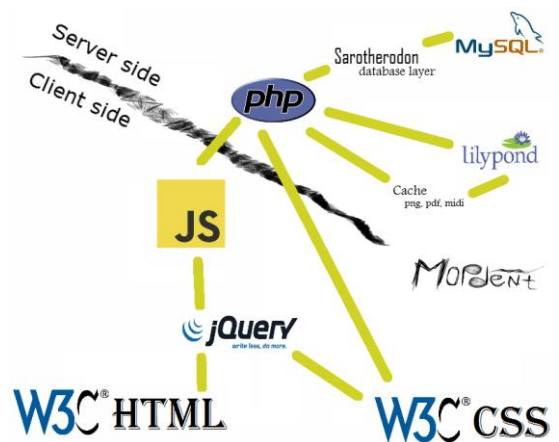


Figure 10: Used systems (extended)

4.2 Application structure from the outside view

4.2.1 Structure of URL addresses is virtual

Via *mod_rewrite* (module of Apache server) all non-existing files are directed to the main script *index.php*, where they are further treated.

As you will see below, every URL node lead in precise point of the application. It likely cannot occur that user enters a virtual address that does not exist -> URLs are friendly.

4.2.2 Pages

Here we provide a list of some pages that browser access. Quick look allows good notion about application structure.

- / - homepage
- /edit/ - new editing new song instance
- /edit/song=*id* – editing song instance
- /edit/song=*id*/ajax/save – user saves the song
What is meant by ‘*ajax*’?
Only result text is requested, no HTML template is called on the output.
For Ajax definition see: Ajax, p 42 .
- /edit/song=*id*/ajax/query/chords – unknown chords tries to be resolved
- /view/ - public songs listing
- /view/song=*id* – song details
- /view/song=*id*/pdf – PDF file is generated
- /view/song=*id*/pdf/chord-staves/ - the chords will be in different stave if possible (see 5.1.4 Export, p 31)

4.3 Login: LoginRadius

We found free login solution. LoginRadius (16) application serves as an intermediary between Mordent and various social networks. Login options of LoginRadius as implemented, are to be seen at Figure 11.

Mordent received unique key which is to be stored in the database. Whenever user returns, they are authenticated via this key.

However, we found a disadvantage: LoginRadius demands details that are not needed. E.g. Facebook user is **asked to reveal their birthday**. Thus, they might decline to log into Mordent because it is weird if music notation software claims sensitive information. For this reason, the login algorithm was changed after a year and LoginRadius is used no more. We use our proper library to login to Facebook, Google and Mozilla Persona¹⁴.

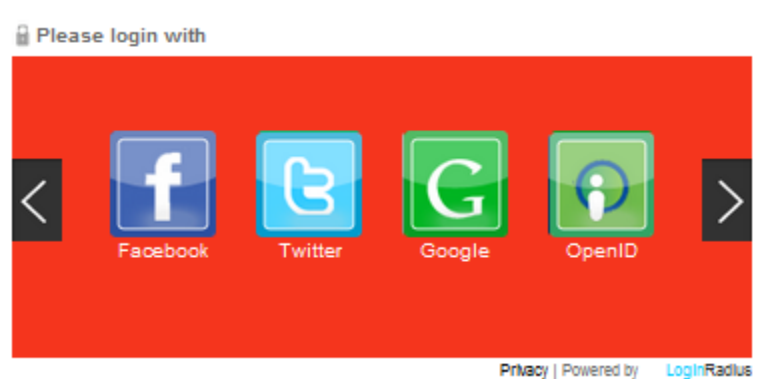


Figure 11: Login options in Mordent via LoginRadius

¹⁴ <http://www.mozilla.org/en-US/persona/>

4.4 Performance boost

4.4.1 Script caching

Even though there is around 30 files of type *.js*, JavaScript code is concatenated into two or three integral files. (Depending on the page viewed.)

Thus, browser has to perform only few download requests. Let's take a quick look onto the Figure 11.





URL	Status	Domain	Size	Timeline
GET javascript.js	200 OK	mordent.cz	333.5 KB	 594ms
GET edit.js	200 OK	mordent.cz	76.1 KB	 329ms
GET jquery.bpopup-0.5.1.r	304 Not Modified	hub.loginradius.com	1.6 KB	 147ms
GET redslider.js	304 Not Modified	hub.loginradius.com	281 B	 294ms
GET jquery.min.js	200 OK	ajax.googleapis.com	29 KB	

Figure 12: JavaScript request duration for editing

In the timeline column, green colour means connecting, purple colour waiting and gray colour data receiving.

- *javascript.js* – The downloading time of the biggest file that contains all plugins (used for instance for better animation) can be split into two parts – about $\frac{3}{4}$ of downloading time was spent on reception of the file itself, while the remaining $\frac{1}{4}$ was wasted on connection. Since *javascript.js* contains multiple files, we save a lot of time which would have been otherwise wasted on connection.
- *edit.js* – contains 9 files
- some files needed for login
- *jquery.min.js* – Row is gray. It means that file has been cached before and there is no need to download it again.

CSS is concatented in a similar way.

In the future, when no more debugging will be needed, the **minification will be turned on** (see Minification of JavaScript, p 42) so that the downloaded code will become smaller (and also little bit protected from theft).

4.4.2 Server caching

Since Lilypond takes several seconds to generate a file, the output is cached. The name of the cached file contains **hash of the contained song**.

Thus, when user comes with direct link to the image, Mordent easily recognizes if the song has been generated before and does not let Lilypond to do the same work twice.

4.5 Source code

4.5.1 Directory structure

- saro - Framework Sarotherodon.
- object - Database objects, the same tables are in database.
 - chord.php
 - song.php

- stave.php
 - user.php
- snippet
 - ‘edit’ – user editing interface and server that saves to database
 - ‘view’ – exporting from database, parsing for Lilypond
 - ‘template’ – html code for homepage, song detail...
 - common files (like CSS)
- design
- plugins – e.g. jQuery
- cache – the files Lilypond generates are stored here

4.5.2 PHP framework Sarotherodon

Sarotherodon is a system of indispensable classes I use in every PHP project.

Database layer

When a PHP application is designed in accordance with object oriented programming principles, the database should likely contain object instances of PHP classes.

Every class extended by database layer:

- is treated as representation of database object,
- its instance represents a database row,
- gets static methods that work over all the database rows.

Virtual structure of URL addresses

Class Fork facilitates the orientation in URL address. Let us explain on the example. Here is the requested URL:

| /view/song=80/png/

With class Fork, every class belongs firmly to the structure. The URL is passed that way:

- ‘view’ – is treated by the main class -> *ViewServer* object is called
- ‘song’ – is treated by *ViewServer* -> gets song ID ‘80’ -> instance of *Song* is created
- ‘png’ – is treated by *ViewServer* -> *LilypondServer* is given the *Song* instance and an instruction to generate PNG file

Without class Fork we would have to check **all the dependencies** manually for every change. For the structure would be held **in both** Apache server and classes, maybe like that:

- ‘view/song=\$1/png/’ is treated by an Apache server -> class *ViewServer* is given the song id and instruction to generate PNG file
- song id and instruction is treated by *ViewServer* -> *LilypondServer* is called

Additional functions

Besides, Sarotherodon facilitates the debugging, effortless caching, HTML forms generating, or image resizing.

4.5.3 PHP objects

- **song.php**
An instance of song checks if it's being edited, its privacy (if logged user has right to view it). Then, it can change its owner, it saves itself...
- **chord.php**
Analyzes unknown chord string.
Chord analysis is on the high level. Not only it identifies complex chords constructs:

| Csus4, C9+, C>7, C5+sus4 Cadd9, C/G

It memorizes user defined non-standard chords and it transposes them up and down when needed.
When user defines an unrecognized non-standard chord construct such as:

| C13+/7*^

chord.php finds the root chord and automatically returns chord variations:

| D13+/7*^, Dis13+/7*^...

- **stave.php**
Secures different stave types (chords, lyrics, melody) are treated in its own right.
- **user.php**
Management of registered, logged, or unknown users.

4.5.4 Database objects

They correspond to PHP objects. Let's take a look about some properties they hold.

- **chord**
owner, song where it has been used, its chord mark, Lilypond representation of its note pitches, time of creation
It is ready to contain guitar tab if implemented in the future.
- **song**
owner, title, author, description, time of creation, cached user plaintext, privacy level
- **stave**
song it belongs to, type (lyrics, melody, chords), user plaintext and Lilypond representation, time of creation
- **user**
login information, name, time of last login, any other information provided via the login site (e-mail)

4.5.5 Server side

Now, when all the objects are described, here are the classes that manipulate these objects and connects them to HTML templates:

- **view_server.php**
Song details, exporting.
- **edit_server.php**
Saving, loading, chord queries.

4.5.6 JavaScript objects

JavaScript objects are used while editing. Let's have a user plaintext.

C7	F	C	F
c c	f f	c c	f f
London's burning, London's burning.			
C	F9	C	F
g	g	a a	g g a a
Fetch the engines, fetch the engines			

It will be distributed into following classes.

- **stave.js**
3 staves: chords ('C7 F...'), melody ('c c ...'), lyrics ('London's burning...')
Every stave contains two rows.
- **row.js**
First of lyrics stave rows is '*London's burning*', second is '*Fetch the engines*'.
Chord stave row contains chords, note stave row contains notes.
- **note.js**
Every note knows its pitch, duration, and plaintext representation including the number of spaces before next note. This is indispensable information, needed e.g. while adding hyphens.
(For adding hyphens see 3.2.3 Bridge between database and output processor / Lyrics and melody, p 18.)
- **chord.js**
Every chord knows its pitches, duration, and plaintext representation including the number of spaces before next chord.

When saving, staves are asked to serialize themselves. Recurrently, every stave asks the same its rows and the rows its elements, so that only one serialized string is sent to the server.

Server creates instance of Stave object for every stave received.

4.5.7 Client side

Let's describe the most important JavaScript classes.

- **directive.js.php**
The only JavaScript file generated by PHP contains backstage variables such as song id, information if user is logged, privacy level, and root directory.
Thus, the variables can be centralized on the server side and there is no need to keep them on two places.
- **engine.js**
Launches textareaManipulation and manages save/load from the server.
Autosaves the project every minute.
- **textareaManipulation.js**
Manages editing, launches analyze of plaintext whenever changed, listens to shortcuts.
- **userCommunication.js**
Being called from anywhere, it popups piece of information:
 - Dialog – full-screen announcement that user must settle before continuing the editing
 - Hint – notification that user can ignore
 - Question – in-offensive dialog

Why in-offensive dialog? Unlike for standard dialogs, user does not have to reply.

When non-standard chord is not recognized, the user is prompted to fill in its pitches.
However, user might be in the middle of writing. Once finished, maybe it turns out that chord is standard after all.
So it would be superfluous to fill in the dialog.

Chapter 5 Result

Mordent has been tested and works in four major browsers Firefox 10 and Google Chrome 15, Opera 12, Safari 5.1 and Internet Explorer 9.

5.1 Description: user experience

We will describe the application from the user point of view.

5.1.1 Homepage

The entry screen is dominated by a vertical column in the middle of the page. This column is divided into big pink cells with general navigation over the site. First cell contains basic information about Mordent.

Other cells allow user to directly create new score or to login. We did not intend to overload the user with information. (From the left side, flying notes begin to spread over the page, slightly rotating.)

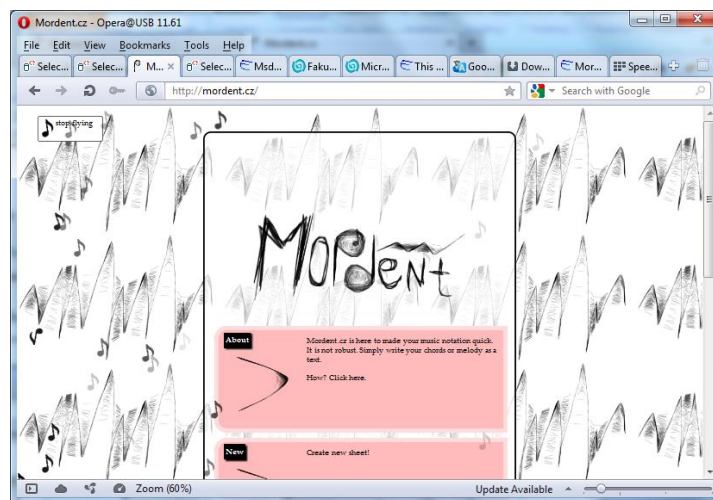


Figure 13: Homepage in Opera 11

5.1.2 Editor

In the beginning, welcome text appears in the screen centre. Mordent informs user about important segments of editing interface.

(Help text may appear again, whenever ‘help’ button in the right menu is clicked.)

On the left side of the page, note the textarea. This is the main input field where user enters the plaintext.

Staves

By default, **plaintext** is splitted into 3 staves of chords, melody, and lyrics.

(If melody or chords are not needed in the song, the staves can be changed through “reorder staves” button on the right side.)

Chords

When user’s plaintext is finished, they may change pitches for individual chords.

On the Figure 15: Editing - chord change we see that chord change field went green when the pitches for chord C7 have been successfully set.

Rhythm

In the Rhythm selection column, we see a bunch of notes. For every melody and chords row, user sets the note durations.

Actual durations are written by numbers in the text field (‘8 8 8 8...’) – all the rows are filled with eighth notes) and represented by the notes above the text field.

Below the text field, other possible rhythms are offered. Mordent tries to predict the most convenient rhythm based on previous experience or estimated length of the row. Offered rhythm becomes active once clicked by mouse or navigated by keyboard shortcuts.

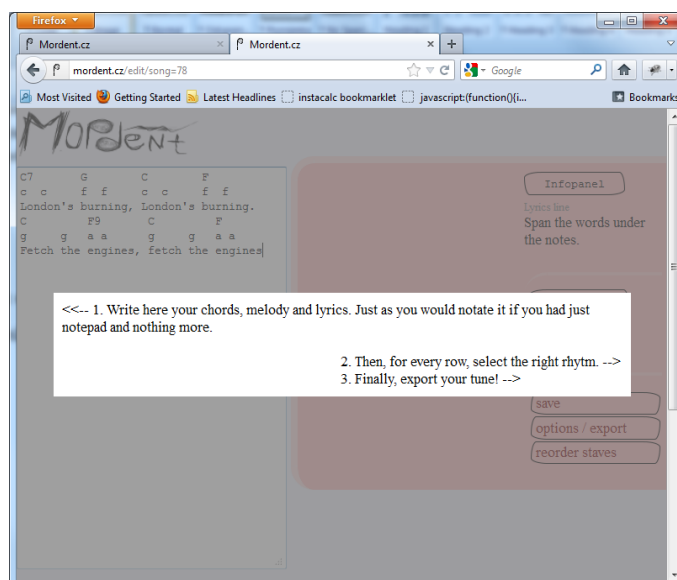


Figure 14: Editing - welcome text

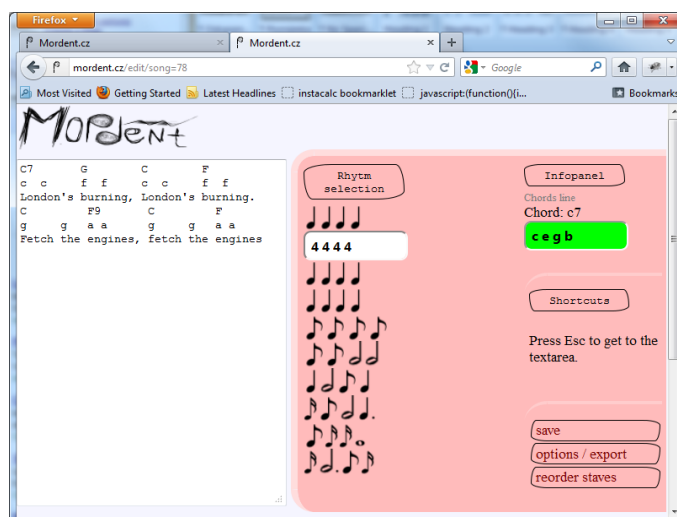


Figure 15: Editing - chord change

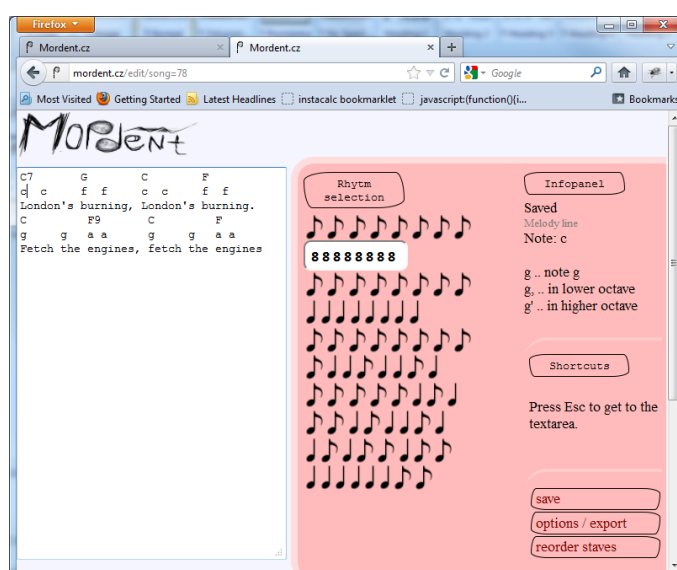


Figure 16: Editing - melody line

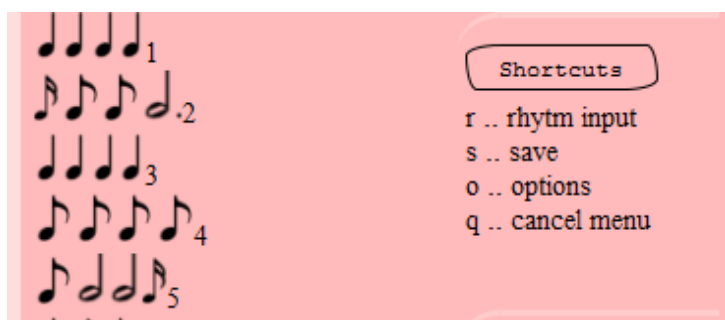


Figure 17: Shortcuts menu - special key pressed

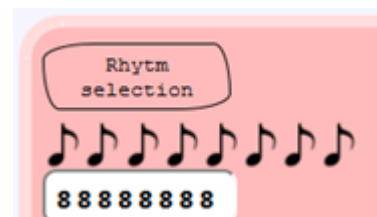


Figure 18: Rhythm selection

Mordent purposely separate the pitch input from the duration input as explained above (see 1.5.4 Plaintext: Natural notation on the computer / Rhythm).

Shortcuts

When special key is pressed, shortcuts panel changes (see the picture), menu appears ('rhythm input, save, options, cancel menu').

Now, if any of the letters marked in menu is pressed, Mordent launch the action associated to the shortcut.

Notice that notes on the left side obtained numbers as well ('1 2 3 4 5') so that user can easily choose rhythm by keyboard.

(I tried to imitate the way shortcuts are implemented in Microsoft Word 2007¹⁵.)

5.1.3 Song detail

The moment user clicks on 'options / export', they come to the song detail page.

Here, piece of information such as song title or song author may be set. As soon as the form is confirmed by OK button, the data are stored into database and all the fields turn green.

Below, export options may be set. Users choose whether they want to display chords in additional stave or what staves are to be exported.

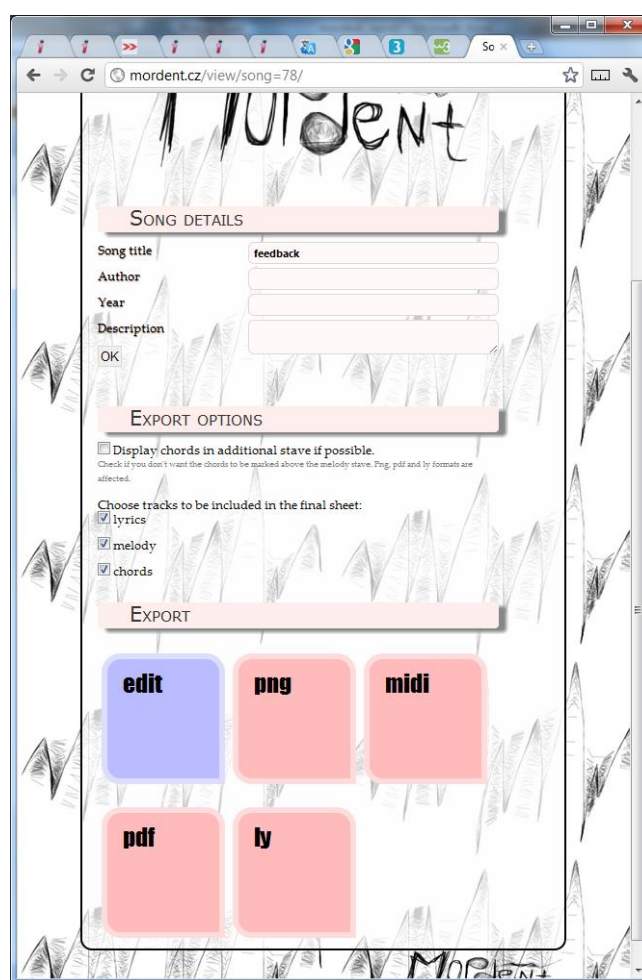


Figure 19: Song details page in Chrome 15
In current version, sharing & privacy options have been added

¹⁵ When Alt is pressed, upper ribbon covers with letters. User can follow these letters deep into menu. Soon, he memorizes even long shortcuts.

Not all the combinations are allowed. If neither melody nor chord stave is passed to the output processor, export is prohibited -> there would be nothing to export.

While a checkbox is clicked, affected export options flashes in order to signalize they are ready.

(E.g. it is not possible to display chord marks in MIDI, since it contains notes only. Thus, the 'Display' option will have no effect over 'midi' button.)

5.1.4 Exported song

Here is an example of Mordent PNG output. One song has been exported with different 'Display chords in additional stave' value.

Export – chord marks

Chords are marked by letters above the melody stave.



Figure 20: Export - chord in marks

Export – chords in stave

Chords are interpreted by its pitches and put in additional stave.



Figure 21: Export - chords in stave

5.2 First version conclusion

The aim of this work on Conventry University was to fulfill the hypothesis of the Proposal sheet submitted in November 2012¹⁶. Let's take a look on the hypothesis and the partial objectives.

5.2.1 Hypothesis

***“Launch** the project Mordent.cz which will be available for everybody. Let the people come and **use it easily** with no difficulties. Let the people write their music by the keyboard (or by the mouse), by the **manner they prefer**.”*

Launch

¹⁶ Accessible also at http://upload.edvard.cz/300com/rejthare_projectProposal_111124.docx

Mordent has been launched. Even though there are still many bugs to treat, everybody can access it and use it.

Use it easily

I tried to design the interface so as there is no trap.

By the first feedback I received, I noticed people are in shock and ask where is the stave, where is the clef and the lines. They consider plaintext notation as something new. I have been disappointed because I reckon there is more plaintext notation on the internet than score.

There is more Chinese speaking people than English speaking people on the globe¹⁷.

In the current version of the application, graphic stave has been added to the editor so that the user sees the graphic presentation of the notes and chords.

Manner they prefer

Since I reckon plaintext is a natural way of notation (see 1.5.4 Plaintext: Natural notation on the computer), I aimed Mordent for it.

People are free to edit their songs offline; they can perform various text transformations (e.g. search and replace) in their favourite text editor. They can digitalize their own texts.

There is no other software that analyzes plaintext notation (see 2.1 Other software on the market, p 7). Mordent.cz is therefore a globally unique service to musicians worldwide.

I implemented shortcuts so that they can use both mouse and keyboard according to their preferences.

5.2.2 Objectives

Buy domain *mordent.cz* and get it a hosting

Done. Domain is hosted at provider savana.cz¹⁸

Let the people say: “It was fast!” Video tutorial placed on YouTube that will document the making of three songs. It will demonstrate the power and speed of song writing. (Pillar: Quick)

Done. I chose CamStudio Portable (17) for screen capturing¹⁹.

Let the people say: “It’s sure for everybody!” The possibility to contribute by developing new keyboard module. Englishman will write “B” whereas German will write “H”. (Pillar: Modular)

Done. There is no need of different keyboard modules. Output processor recognizes both H and B. Mordent analyzes the chords.

Let the people say: “There is no obstruction!” Versioning & importing from standard music formats (like MIDI). They create a tune without logging in. The tune will be autosaved and public unless they decide to register a username. (Pillar: Friendly)

¹⁷ This is just an illustrative comment, permit me not to prove the evidence.

¹⁸ CPU 400 Mhz; 384 MB RAM

¹⁹ <http://mordent.cz/outcomes>

Half done. Autosave has been implemented and song is public if user does not log in, as requested. Import has not been implemented yet. Lilypond contains module that parses MIDI, thus this would not be problem. Unfortunately, other features came ahead and MIDI import remains on the task list among other enhancements.

Let the people say: *"It has been really done!"* The ability to download a scoresheet PDF.

Done.

Let the people say: *"You can go there!"* Facebook app that allow the people to login with their Facebook account.

Done. People can log in not only with their Facebook account, but also with G+, OpenID, Twitter and Wordpress.

Let the people say: *"Just as I dreamt!"* GUI which make sense and guide them through all the time.

Done? The class userCommunication (see 4.5.7 Client side, p 26) facilitates the type hinting in Mordent. Every single change of caret position in textarea summons appropriate help text.

Mordent was the author's dream that became true. He will certainly use it in order to digitalize both his old and new songs. He cannot but hope that others have the same dream.

Let the people say: *"Let's use it again!"* Collaborative editing possibility.

Half done. Collaborative editing is enabled. If song is not locked by other editing user, and if it is not private, other user can make changes.

On the task list, there is a possibility of creating secret *bit.ly* links. With such link, anyone could either view or edit the song.

5.2.3 Outcomes

Mordent.cz web application as described

Done. To be seen at <http://mordent.cz/outcomes>

3x YouTube videos, or tutorial

Done. Explained above.

3x PDFs, or examples

Done. To be seen at <http://mordent.cz/outcomes>

Facebook application

Done. Facebook login works online.

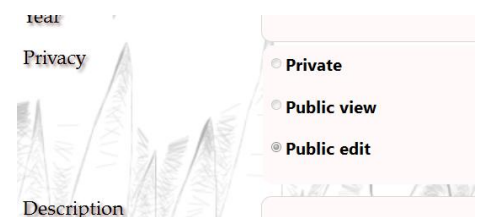


Figure 22: Privacy settings

5.3 Further development

The developed went beyond the original assignment. The testing within the CTU course A7B36TUR (18) brought some notions that were further implemented. Since this work is submitted one year after the original version had been launched, we introduce the list of changes that were realized.

5.3.1 User testing

We used screener form to choose five participants. Finally, we got 5 musicians between 19 – 30 with the active ability of english language speaking. They were given the 9 step scenario that simulated the process of

creating a simple music score – they had to find videotutorial on the mainpage, edit the song UseCase 1, change its metadata and save it, display the help text, change the pitches of a chord, create new chord, change playback options, rhythm of a row, and eventually generate output PDF file.

Behind the wall, few moderators supervised the course of the testing and all the notable moment were logged into a file. Besides, the participant's screen and face were recorded by a webcam.

All the participants fulfilled the tasks given. Sometimes, they had problems with orientation on the screen, they had to be told how to change a chord, and they complained about the note playback. Their comments contributed to changes stated in the next paragraph, the original testing and detailed results can be accessed via the sources (18).

5.3.2 Changes

Editor

The main page template has been changed to be compatible with mobile devices. The amount of space taken by the control elements were reduced so all important information can be viewed without scrolling, as seen at the image below.

Since the first version of the application, graphic staff has been added in the upper part of the screen. It displays both melody and chords, highlights the position of text cursor in the textarea, and shows the note which is being played. Menu buttons were placed above the graphic staff in favour of many new buttons that perform additional auxiliary functionality. E.g. the user can move in the graphic staff by buttons “go left”, “go right” (or via its one-key keyboard shortcuts “j”, “l”) or they can select some notes or chords and automatically transpose them up or down.

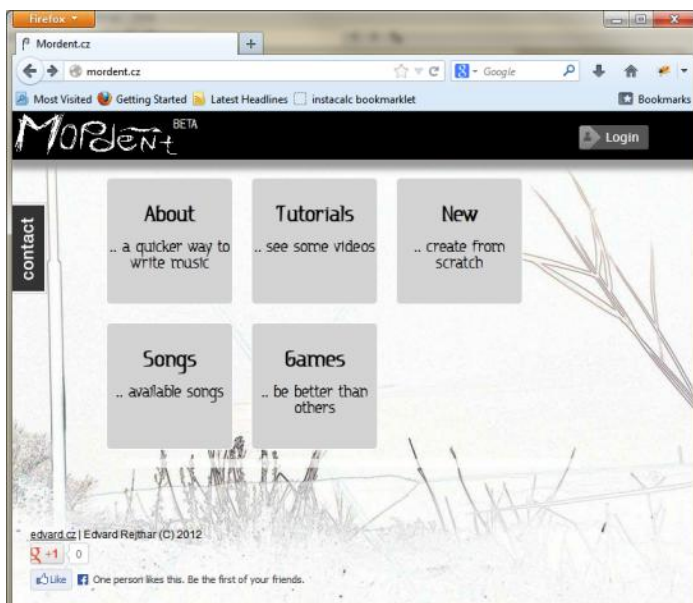


Figure 24: Main page button layout

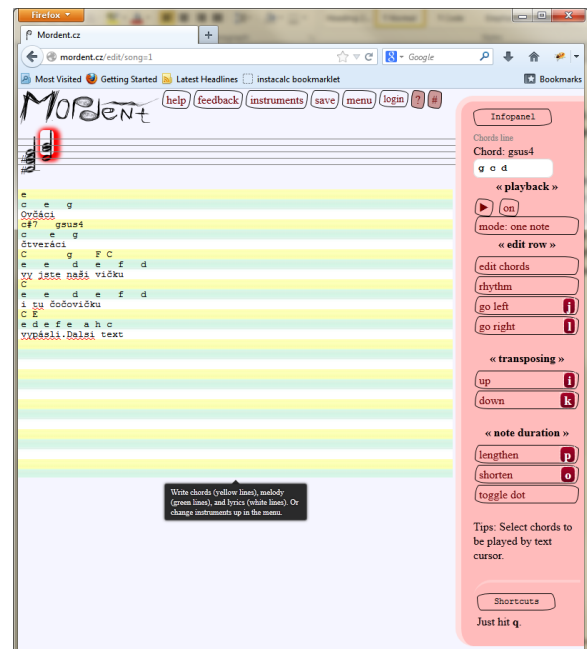


Figure 23: New editor functionality – menu buttons, highlighted chord in graphic staff

Games

Music games were added that form completely new functionality branch in Mordent. Apart from the music editor, Mordent now contains six music games that allow user to train their hearing ability capabilities and their music theory knowledge.

As seen at the Figure 26, the game engine asks the user some question they have to respond. They are given an answer panel of buttons – in this case, it is green and contains pitches. Below, score is displayed and time is elapsing. In our case, user just got +0.25 points so there is big +0.25 number fading on the left side. When user answers, current tone pitch is displayed on the graphic stave and then another question is asked. At the Figure 25, “Decomposition chord” is being played. In this game, the player has to input all the pitches the chord contains. It is a useful training for guitar players that improve their music theory notion.

On the each of the games’ right side, ranking is showed so the player see at each moment how many points they have in comparison with other players. They see 3 top players of the game, then 3 players above, themselves and 3 players below in the ranking ~ 10 items. (The algorithm is treated so that if the player gets at the top of the ranking, they see only 3 players below ~ 4 items.)

Further, there is contact button that have been implemented on every Mordent page. With its help, the visitor can easily feedback their opinion or submit a bug encountered.

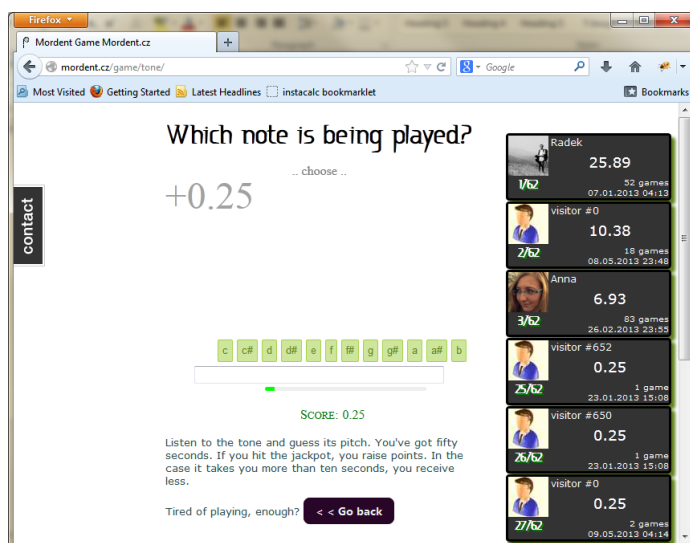


Figure 26: "Single tone" game

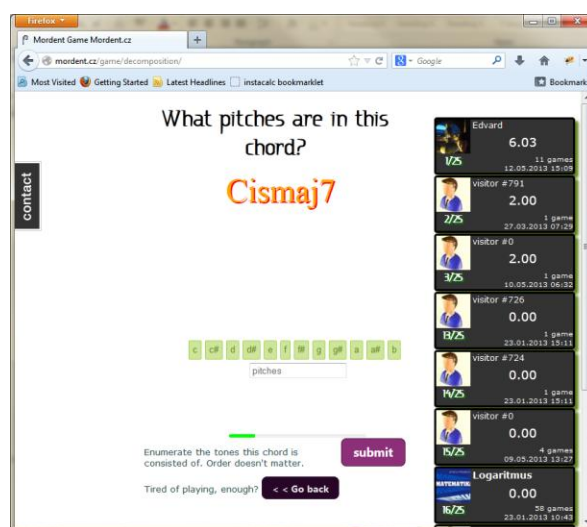


Figure 25: "Decomposition chord" game

5.3.3 Future steps

Since the first version of the application, many tasks has been accomplished. The most important was the graphic stave (five horizontal lines) implementation and playing through. Now, user is able to hear the music as they type. But few interesting tasks are still listed: Czech language, MIDI import and guitar tablature export.

Chapter 6 Gained experience

The author notes might be useful to anybody who is going to develop similar application.

6.1 Hard decisions the author made

6.1.1 Lilypond chord input

As seen in the Figure 8: Demonstration of a Czech traditional song (p 12), chord mark does not fit above the notes. Why? Is not Lilypond able to proceed chords correctly?

It has been said that Mordent has to accept even non-standard chords (3.2.2 Handy chords, p 17). But Lilypond accepts only standard chords, G^{\wedge} would not be accepted as a valid chord mark.

So that the author has been searching for another feature of Lilypond that would permit him to write some text above melody. Now, I am using Lilypond *Text marks* (19) and not *Chords* for chord displaying. And since original purpose of *Text marks* is something different, the layout does not fit precisely.

6.1.2 Temporary users

It has been said (3.2.6 Login, p 19) that Mordent does not impose user to register. The author had to implement one of these two options then:

- **Do not insert** user in the database **until registered** and logged
 - Mordent application have to treat two states: (1) logged user in the database, (2) unknown user not in the database
- **Insert** unknown user in the database as if he is logged?
 - >> Lot of garbage in the database
 - Add column "Temporary user = yes" to the user table and delete the database by a *cron*²⁰ request?

Eventually, the author implemented that unknown user is immediately created in database as if he is logged. If they log, their account **is merged** with their previous login account.

When an unknown user comes, it tries to recycle first some unused²¹ users rather than create new row.

6.2 Troubles encountered

6.2.1 JS

Keypress in Chrome

I wanted ESCAPE key ("Esc") to be the special key that launches shortcuts.

However, there are some limitations. Take a look on some facts:

- Keypress event can cancel key by returning false; so that letter is not written
- Keydown event cannot handle Esc in Chrome

²⁰ Periodical instruction, launched by server in a certain time.

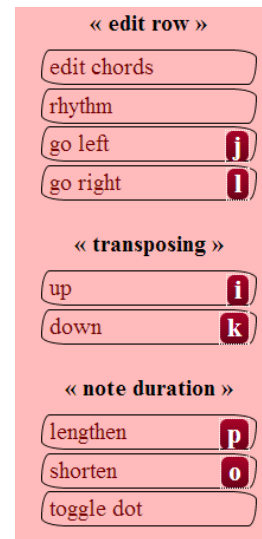
²¹ Unused user: Its creation time in database is older than a day and they do not have permanent-login cookie key. (So they cannot log in again.)

- (arrow keys are treated differently as well)
- Unlike in Chrome 15, while Esc pressed in Chrome 17, input field defocuses
- It is not polite to remap F1-12 keys, reserved for other actions in browser

Finally, I elected 'q' for the special key. It will pose problems when writing in languages with often occurrence of 'q'. Therefore, a functionality will be implemented that allows user to change the special key or to switch off the entire shortcut functionality.

Besides, some one-key shortcuts have been implemented in the current version, as seen at the side image. If the user needs to write these reserved letters, the one-key shortcuts functionality can be turned off in the menu.

Figure 27: One key shortcuts in the current version



JavaScript Closure

Shortly said, *scope* (20) means that the variable is recognized in its local context only (current method or function), only around itself.

Besides, JavaScript has *closure* (21), it means that the variable can be accessed in children method too.

Since I did not know about the existence of closure and I did not declare local variables with the keyword *var*, the iterator in the cycle influenced the iterator of the same name in the parent method.

```
for(i = 0; i < ...; i++)
  childMethod(); //goes to the children method

childMethod = function() {
  ➔ //in the children method
    for(i = 0; i < ...; i++){
      //outer i is influenced
    }
}
```

For cycle

While traversing:

```
| for(var i in rhythmA)
```

I cannot splice the traversed array on the fly. When I try it, the next value is skipped. To achieve that, the ordinary cycle has to be used:

```
| for(var i = 0; i < rhythmA.length; i++)
```

Browser restart

During debugging, a JavaScript method returned once an empty array. Another time under the same conditions, it returned an infinite-length array of an infinite depth. And browser crashed.

It was quite difficult to solve such a problem. I have never fully understood what caused it.

Object assignment

```
get = function (i) {  
    return array[i]; //array of object, accessible via global  
    score  
}  
  
get(0) = new Object(); //array[0] is not affected  
array[0] = new Object(); //here is array[0] affected
```

Skilled programmer may despise this kind of problem. But it took me several minutes to figure out where the problem was.

String addition

```
i = 1; i2 = '2';  
result = i+i2;
```

Be warned, *result* is not 3. It is 12 because variables are treated as string, not as integer.

6.2.2 MySQL

Column = 0

```
select * from chord where `plaintext` = 0
```

SQL query above returned rows where plaintext = 'fa' as seen at the picture. I never realized why.

SQL command							
<pre>select * from chord where plaintext = 0</pre>							
id	title	user_id	song_id	plaintext	pitches	tab	timestamp
10		0	1	fa	c e g ais		2012-03-18 19:58:52
32		0	1	faa	d a h		2012-03-18 19:58:52
2 rows (0.000 s) Edit , EXPLAIN , Export							
<pre>select * from chord where plaintext = 0</pre>							

Figure 28: Strange behaviour of SQL command

6.2.3 Browser

Textarea '\n\r' or '\n'

New line sign in HTML textarea is sometimes \n\r, sometimes \n. I did not concentrate whether this problem is caused when pasting into textarea or when did it occur.

It is sufficient to be aware of it when parsing the textarea string. (The new line sign should differ according to operating system.)

CSS3 support

Internet Explorer 8 which is installed on every computer in University campus, is not able to treat the CSS3 properties well.

If you look on Figure 29: Homepage in IE 8, you will see that unlike in other browsers there are no round borders. And the flying notes look thick and ugly.

I use the border-radius property which is not supported in Internet Explorer browser in version 8 and below (22). The similar problem is with box-shadow property.

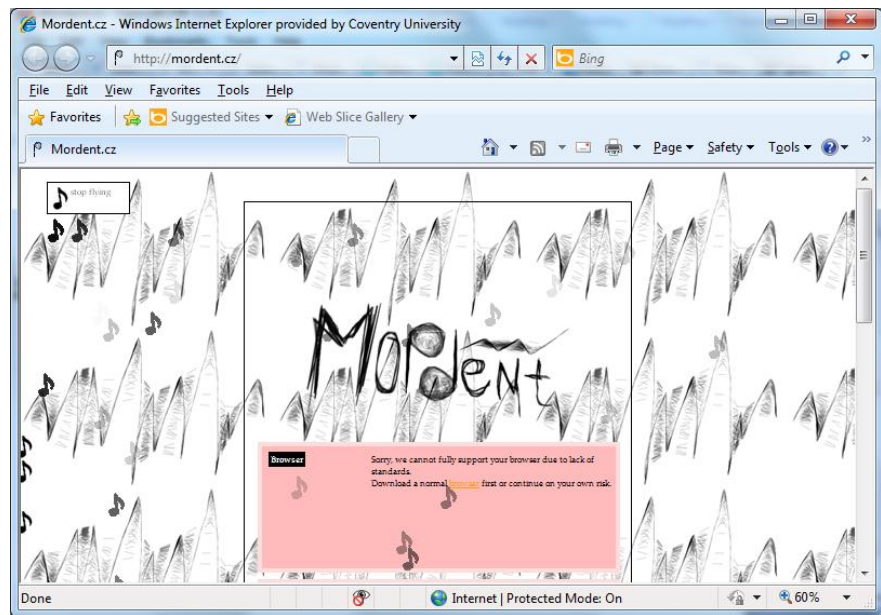


Figure 29: Homepage in IE 8

6.3 Lessoncs learned

PHPDoc

I like the type hinting functionality in the IDE: When I begin to write the variable name, its methods are offered to me. Unfortunately, PHP type juggling (23) makes type hinting difficult.

Previously, I added an ‘instanceof’ expression to indicate the ClassType:

```
| if($variableName instanceof ClassType)
```

Now, instead of adding superfluous condition, I use PHPDoc notation:

```
| /* @var $variableName ClassType */
```

Declaration order

While traversing the \$composant array, first iterated key will be “nona”:

```
| $composant["tercie"] = $composant["sexta"] = $composant["septima"]
| = $composant["nona"] = 0;
```

6.4 History log

First version of Mordent application, described in this work, has been written in circa 130 hours. Here is some extraction from my personal logbook that follows the order of steps taken to launch the core .

5h sketching on papers
5h still sketches on papers
4h liquid horizontal boxmodel; js classes
9h js classes; userCommunication, parse chords preparation
4h Lilypond installation, ly->pdf
9h textarea -> DB -> PDF
12h caret position over the note, load, liquid textarea, rhythm algorithm
15h rhythms working (prediction, cache, server, print); lyrics
8,5h Lilypond trims PNG; 4h flying notes + homepage; 2h chord parsing
3h user can change chord, user can save
6,5h chords are loaded from server; parsed; edit interface design; login research
14h login implementation; song details template
5h debugging, shortcuts
3h options, asking server debugging
2,5h rhythm prediction
3h display chords in additional stave
4h debugging
14h debugging

Annexes

References

1. M-Audio Keystation Mini 32. *Reg Hardware*. [Online]
http://www.reghardware.com/2011/09/19/geek_treat_of_the_week_m_audio_keystation_mini_32/print.html.
2. Finale. [Online] <http://www.finalemusic.com>.
3. Finale SmartMusic. [Online] <http://www.smartmusic.com/>.
4. Finale PrintMusic. *Music Notation Software Review*. [Online] <http://www.music-production-software.com/product/finale-printmusic.html>.
5. MakeMusic. *Finale vs. Sibelius*. [Online] [Cited: 05 01, 2013.]
<https://store.makemusic.com/Store/Switch.aspx>.
6. Sibelius store. [Online] http://www.sibelius.com/buy/gb_edu_single.html.
7. NoteWorthy Composer. [Online] <http://www.noteworthysoftware.com/>.
8. Midisoft Record Session Program. *Greatest Metallica Midi Page!* [Online] Feb 28, 2011. [Cited: Apr 2, 2012.] <http://metallicamidis.blogspot.co.uk/2011/02/midisoft-record-session-program.html>.
9. *Noteflight.com*. [Online] <http://www.noteflight.com/>.
10. *Lilypond.org*. [Online] <http://lilypond.org/>.
11. *Denemo*. [Online] <http://www.denemo.org>.
12. Native and foreign format. *Wikipedia*. [Online] http://en.wikipedia.org/wiki/Native_and_foreign_format.
13. **Cole, Richard**. Table of Musical Ornaments. *Virginia Tech Multimedia Music Dictionary*. [Online]
<http://www.music.vt.edu/musicdictionary/appendix/ornaments/ornaments.html>.
14. F. L. Bauer. *PediaView*. [Online] http://pediaview.com/openpedia/F._L._Bauer.
15. **Lavin, Peter**. *Object Oriented PHP*. San Francisco, CA, USA : No Starch Press, Incorporated, 2006.
9781593271275.
16. *LoginRadius: Social sharing*. [Online] <https://www.loginradius.com/>.
17. **Whitty, Bryce**. Repair Tool of the Week: CamStudio Portable. *TechNibble.com*. [Online]
<http://www.technibble.com/repair-tool-of-the-week-camstudio-portable/>.
18. **Jakub Kopřiva, Jakub Poukar, Edvard Rejthar, Václav Balon**. Testování uživatelských rozhraní. *HCI Semestrálky*. [Online]
http://hcisemestralky.felk.cvut.cz/system/assets/2777/original/Semestralni_prace_C1_-_Testov_n_webov_aplikace.pdf.

19. Writing text / Text marks. *Lilypond Documentation*. [Online]
<http://lilypond.org/doc/v2.12/Documentation/user/lilypond/Writing-text#Text-marks>.
20. Scope (computer science). *Wikipedia*. [Online] [http://en.wikipedia.org/wiki/Scope_\(computer_science\)](http://en.wikipedia.org/wiki/Scope_(computer_science)).
21. More closure examples. *JavaScript Kit*. [Online]
<http://www.javascriptkit.com/javatutors/closures2.shtml>.
22. CSS3 border-radius Property. *w3schools*. [Online] http://www.w3schools.com/cssref/css3_pr_border-radius.asp.
23. Type Juggling. *PHP*. [Online] <http://php.net/manual/en/language.types.type-juggling.php>.
24. Build Software Faster with Less Stress. *Assembla*. [Online] <http://www.assembla.com>.
25. An Identity System for the Web. *Mozilla Persona*. [Online] <http://www.mozilla.org/en-US/persona/>.

Definitions

WYSIWIG

“What you see is what you get” editing. Editor imitates the appearance of the final result.

If you need to indicate bold text in plaintext editor, you may do it e.g. this way (HTML):

```
| This text is <b>bold</b>
```

Whereas in WYSIWIG editor, you see bold text immediately.

```
| This text is bold
```

MIDI

MIDI is a digital format which preserves data that might be turned into sound signals. MIDI files can be played in most software players. Besides, music instruments may output the MIDI data through cable.

Ajax

Ajax is technique which allows browser to communicate with the server asynchronously. As a result, not entire page has to be refreshed when sending and receiving data.

Minification of JavaScript

During the process of minification, all characters that are not required, are removed. The variable names and methods are renamed so that the result output is as small as possible.

Computer launches the new code without obstacles but human cannot read it anymore.

Contents of the CD

- this document in .docx (rejthedv.docx)
- this document in .pdf (rejthedv.pdf)
- source codes (mordent_current.zip)
- CTU official assignment (rejthedv_zadani.jpg)
- original Coventry report (coventry_report_120326.docx)
- Coventry Project proposal sheet (coventry_projectProposal_111124.docx)