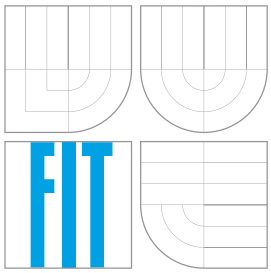
BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ



FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

WEB EDITOR OF MUSICXML FILES

WEBOVÝ EDITOR SOUBORŮ FORMÁTU MUSICXML

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR
AUTOR PRÁCE
TOMÁŠ HUDZIEC

SUPERVISOR
VEDOUCÍ PRÁCE
RNDr. MAREK RYCHLÝ, Ph.D.

BRNO 2016

Bachelor Project Specification/18587/2015/xhudzi01

Brno University of Technology - Faculty of Information Technology

Department of Information Systems — Academic year 2015/2016

Bachelor Project Specification

For: **Hudziec Tomáš**
Branch of study: Information Technology
Title: **Web Editor of MusicXML Files**
Category: Information Systems

Instructions for project work:

1. Study the MusicXML format structure and music notation standards. Get familiar with the process of designing and developing a reactive HTML5 web application.
2. Design an online music notation editor of the MusicXML formatted documents as a HTML5 web application. The editor will be able to load and save MusicXML files and to render their content in a standard graphical notation. A user must be able to modify the content of the MusicXML file using the standard graphical notation in a graphical designer provided by the editor.
3. After agreement with the supervisor, implement the editor. Perform testing and provide documentation of the implementation. Publish the implementation as an open-source at GitHub.
4. Evaluate the results and discuss further extensions and future work.

Basic references:

- FAIN, Yakov, Victor RASPUTNIS, Anatole TARTAKOVSKY, Viktor GAMOV a Sharon WILKEY. Enterprise web development: building HTML5 applications : from desktop to mobile. First edition. Sebastopol, California: OReilly Media, Inc, 2014, xxviii, 609 pages. ISBN 978-144-9356-811.
- FRAIN, Ben. Responsive web design with HTML5 and CSS3: learn responsive design using HTML5 a CSS3 to adapt websites to any browser or screen size. Birmingham: Pack Publishing, c2012, vi, 305 s. ISBN 978-1-84969-318-9.

Requirements for the first semester:
No requirements.

Detailed formal specifications can be found at http://www.fit.vutbr.cz/info/szz/

The Bachelor Thesis must define its purpose, describe a current state of the art, introduce the theoretical and technical background relevant to the problems solved, and specify what parts have been used from earlier projects or have been taken over from other sources.

Each student will hand-in printed as well as electronic versions of the technical report, an electronic version of the complete program documentation, program source files, and a functional hardware prototype sample if desired. The information in electronic form will be stored on a standard non-rewritable medium (CD-R, DVD-R, etc.) in formats common at the FIT. In order to allow regular handling, the medium will be securely attached to the printed report.

Supervisor: **Rychlý Marek, RNDr., Ph.D.**, DIFS FIT BUT
Beginning of work: November 1, 2015
Date of delivery: May 18, 2016

L.S.

Dušan Kolář
Associate Professor and Head of Department

Abstract

This thesis describes implementation of web editor of MusicXML files. Structure of MusicXML format is studied and described. Implementation is done using HTML5, CSS and JavaScript. For notes rendering is used open-source JavaScript library VexFlow. Graphical music score is outputted into SVG, which is part of the web page. Due to complexity of MusicXML format, only its subset is supported. Solution provides loading or creating of music notation, editing it with computer mouse and saving result into a file.

Abstrakt

Tato práce popisuje realizaci webového editoru hudební notace uložené ve formátu MusicXML. Je nastudována a popsána struktura formátu, implementace je provedena v jazyce HTML5, CSS a JavaScript. Noty se vykreslují pomocí open-source knihovny VexFlow do vektorového obrazu SVG, který je součástí webové stránky. Vzhledem k rozsáhlosti specifikace formátu MusicXML je podporována jeho podmnožina. Řešení umožňuje uživateli nahrát/vytvořit a upravovat notový zápis pomocí počítačové myši a výsledek posléze uložit do souboru.

Keywords

MusicXML, editor, web application, JavaScript, VexFlow, SVG, music notation

Klíčová slova

MusicXML, editor, webová aplikace, JavaScript, VexFlow, SVG, hudební notace

Reference

HUDZIEC, Tomáš. *Web editor of MusicXML files.* Brno, 2016. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Rychlý Marek.

Web editor of MusicXML files

Declaration

I declare that this bachelor thesis was developed independently under the leadership of RNDr. Marek Rychlý, Ph.D. Additional information provided to me Mgr. Štefan Bunčiak. I stated all literature and publications, which I used.

.....
Tomáš Hudziec
May 18, 2016

Acknowledgements

I would like to thank Mgr. Štefan Bunčiak for his feedback and RNDr. Marek Rychlý, Ph.D. for his supervision of my thesis. I would also like to thank my friends and my family for support and encouragement.

Contents

List of Figures

Chapter 1

Introduction

Music is essential part of human life. It encourages, calms, inspires, and speaks to human emotions in many other ways. For centuries music was passed on from generation to generation in oral form. Later music began to be preserved on paper. Music notation have made great progress till now.

With an advent of digital technologies computers began to be used for preserving music notation. Many programs for working with music have been developed, many of them are for very different purposes, such as music playback, optical music recognition, music printing, musical databases and more. Lots of programs designed its own format for its purposes. Many different formats cause problems in musical data interchange between programs.

MusicXML format was designed to be an interchange format and solve this problems. Task of this thesis is to implement Web MusicXML Editor.

Chapter 2 introduces into theory of music and its notation. It defines "music" for purposes of this thesis, briefly looks into history, explains basics of modern music notation, and provides preview of computer music formats.

In chapter 3 format MusicXML is studied. Also there are described technologies for development of reactive HTML5 web application.

Chapter 4 provides results of research on existing solutions. There are shown features of solutions both in textual and graphic form.

Application design is included in chapter 5. There are viewed aspects of music notation editing and requirements on such editor. Also application graphical interface is introduced.

Follows chapter on implementation with number 6. Chapter focuses on used technologies, and way how they are used in project. It also covers description of application architecture and models of music score.

Chapter 7 gives information about application working in different browsers. There are provided solutions for revealed shortcomings.

The last chapter 8 concludes work done on this project and describes planned future steps and extensions.

Chapter 2

Music and its notation

To understand principles of music notation used in this work, this chapter defines concept of music and describes basics of music notation. It is important for this thesis, because editor must respect rules, which applies to music. The first and the third sections defines music and explains Common Music Notation (CMN). Second section briefly portrays history and evolution of music notation. In contrast of the history, the last section is dedicated to ways in which computers preserve notation of music.

2.1 Definition of "music"

"The word 'music' means many different things – for example, 'a song,' 'the idea of a song,' 'all music, in general,' or 'pitched events with durations.' "[17] This thesis for its purposes defines music as a structured data:

> Notes – the smallest music elements – are objects with pitch and duration[1]. These notes are grouped into voices, voices into measures(there can be up to two voices in a measure). Sequence of measures forms a part – it is music track for one instrument. Whole music score, if it has multiple instruments, consists of several parts.

This definition comes from how MusicXML format encodes music score. Format itself will be described in next chapter.

2.2 History of modern music notation

Roots of modern[2] music notation can be found in ancient Greece, where people for music notation used letters of alphabet placed above text syllables. Each note had its own letter of alphabet, which is the same concept as nowadays.

This way of preserving music seems to be forgotten in medieval age, when signs called *neumes* had been used for character of melody. Neumes were notated also above text of song, and offered only "general idea of the musical outline". [21, p. 3] They did not represent particular pitch of tone, with its position they showed only rise or fall of melody. Singers already had to know melody of song, it was not possible to recall the melody only from neumes.

[1] Notes can have also other attributes, such as dot, accidental, tie or slur, lyric and more.

[2] By modern music notation is meant western or also called common music notation.

Lines for neumes appeared later, at first one line, which represented absolute pitch of F staff, example is shown on figure 2.1. After there were two lines with different colour and letters F and C assigned to them. These lines were ancestors of contemporary staff, as we know it nowadays. [21, p. 4] Modern clefs in music evolved from these letters assigned to lines. Process of such evolution shows figure 2.2.

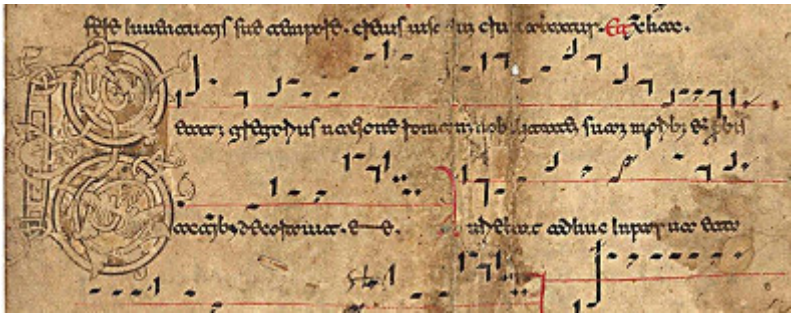


Figure 2.1: Beneventan neumes on a single line red F-staff. Montecassino, Italy, 2nd half of 12th century.[4]

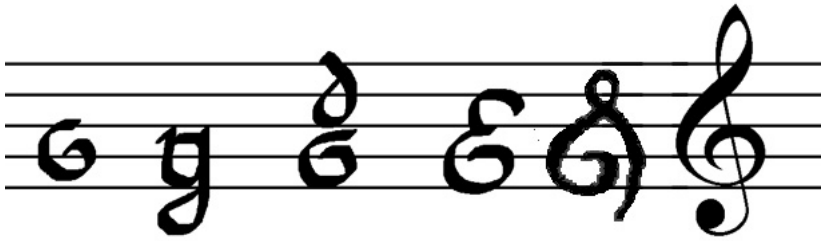


Figure 2.2: Evolution of treble clef.[6] "The treble clef is stylized G, that curls around the second line on the staff." [17, p. 15]

From neumes to notes as we know them today there must have been done even more changes. However, further studying of history of music notation is be behind of scope of this thesis. Interested reader can get more information in work focused on such topic [21].

This section have provided the basic idea of roots of modern music notation and what influenced way how it looks like.

2.3 Common Western music notation

From technical perspective, common music notation is a "graph of pitch against time".[17] As x(time) axis we can imagine a musical *stave* – usually five horizontal lines. Notes are placed on lines or between them.

Pitch (frequency) of note is proportionately determined by its y position and *clef*, which is sign placed in the beginning of stave. Notes are named by letters of alphabet, from A to G in order C, D, E, F, G, A, B. Note's pitch can be altered by one or two *half-tones* up or

[4]Image taken from http://schoyencollection.com/music-notation/beneventan-neumes/antiphonal-beneventan-neumes-ms-1681

[6]Image is licensed with CC BY-SA 3.0 and available on https://commons.wikimedia.org/w/index.php?curid=687950

down by assigning an *accidental* to it. Possible accidentals are: *double-flat* 𝄫, *flat* ♭, *sharp* ♯ and *double-sharp* 𝄪. *Natural* ♮ discards previous accidental.

Note's duration is determined by its shape. There is whole note 𝅝, and shorter notes are its fractions: half note 𝅗𝅥, quarter note ♩, 8th note ♪ and 16th note 𝅘𝅥𝅯.

On figure 2.3 are shown some more advanced elements of music notation. The leftmost symbol on the stave is treble clef. As was mentioned at figure 2.2, it evolved from letter G and has something in common with second line from bottom of the stave. It tells, that note on the second line of the stave has name 'G'. According to the order of the notes, one step above G is note A, then note B and so on.

Sharp on the fifth stave line is the *key signature*. On the fifth line there is note F. It means, that every F note on the stave is implicitly raised a half-step.

Two '4' ciphers on the stave represents *time signature*. Vertical line across the five line stave is called *barline*. It defines boundaries of *bar*, or equivalently *measure*. Time signature is a fraction. "Value of its denominator declares the basic unit of time in measure, and the numerator indicates how many units of time are in a measure. In example from figure 2.3 there are 4 units in a measure, and quarter note is the basic unit of time." [17, p. 15]

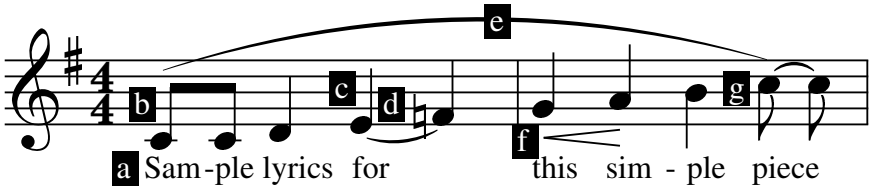


Figure 2.3: *This example demonstrates more advanced notation elements. From right to left, they are: a) lyrics beneath the notes b) a beamed pair of eighth notes c) a slurred pair of quarter notes, indicating that the notes should be blended together as much as possible d) an "accidental" natural in front of an F, which cancels the automatic sharp implied by the key signature e) a phrase mark, indicating that all the notes are to be performed as one "phrase" (perhaps in one breath) f) a crescendo, instructing the performer to increase the volume g) two tied eighth notes, which are performed as one note with the combined time value of both.* Figure and description taken from [17, p. 16].

2.4 Computer music notation

With an advent of digital technologies computers began to be used for preserving music notation. Many format for such task appeared through the computers history.

Very popular and widely used is MIDI format. It is designed for playing music by sequencers, it represents notes as numbered events with pitch in time. However, it is not suitable for music score printing, because it is focused on how music sound, not appearance.

Format called NIFF represents music using graphical format. It works well with scanning programs, but it is not suited for sequencing and musical databases.

Standard Music Description Language (SMDL) was attempt to generalize and support all possible music information from the past, present and future. The standard was too complex and hard to implement, so no software adopted it.

There are many more other formats, which are less significant, just to mention some of them: TEX-based formats for music typesetting – MuTEX, MusicTEX, MusiXTEX, GNU

Lilypond; XML based formats: MNML, MusicML, MusiXML, MEI, WEDELMUSIC and many more.

MusicXML, as format studied in this thesis, emerges at the turn of millennium and soon it will become standard for computer music notating. It will be described in the next chapter.

Chapter 3

Analysis of technologies

This chapter describes analysis of technologies which were studied for development of Web Editor of MusicXML Files.

3.1 MusicXML

This section provides brief introduction into MusicXML format.

3.1.1 Format origin

MusicXML format was developed by Recordare LLC in 2000. It has become an industry standard for music notation. It is mostly used as an interchange format between many programs. Today more than 200 applications include MusicXML format support. In 2015 development of MusicXML format has been transferred to World Wide Web Consortium (W3C), which may move MusicXML development forward in a more open process[1].

MusicXML is designed to represent musical scores, specifically common western musical notation from the 17th century onwards. It is designed as an interchange format for notation, analysis, retrieval, and performance applications. MusicXML is based on formats MuseData and Humdrum, which it adapts to XML. [6]

3.1.2 Structure

Humdrum explicitly represents the two-dimensional nature of musical scores by a 2-D layout notation. There are two types of events(notes in this case):

1. Successive events
    - in one part notes in one part, which go one after one
2. Concurrent events
    - corresponding notes between several parts, which sound in the same time

On figure 3.1 with multipart music are shown such types of events.

[1] According to http://musicxml.com/makemusic-transfers-musicxml-development-to-w3c

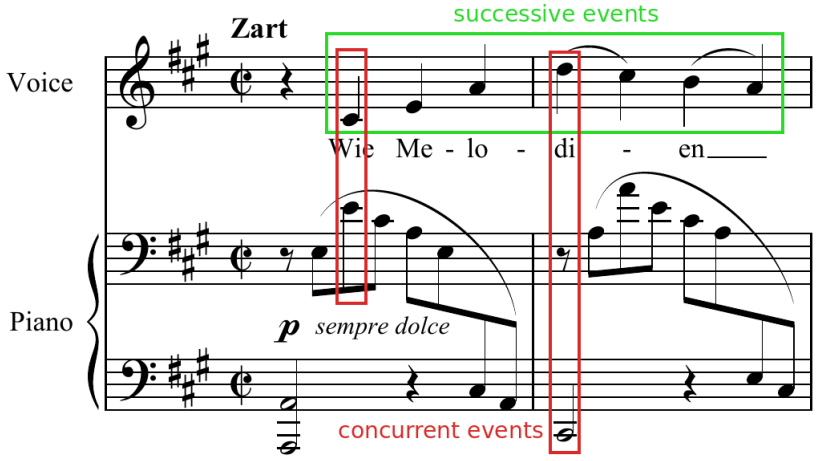


Figure 3.1: Example of common Western music notation from music piece Wie Melodien zieht es mir by J. Brahms and K. Groth.

Because xml is format of hierarchical structure, it is impossible to directly represent such 2-D layout. Therefore there exist two MusicXML structure definitions: `score-partwise` and `score-timewise`. "In the timewise score the measure is the primary element and parts are contained within each measure. In the partwise score the part is the primary element and measures are contained within each part." [7, p. 4] Between these two organizations exist conversions through XSLT stylesheets. In this work we will deal with `score-partwise` arrangement.

On figure 3.2 is shown, how data in `score-partwise` format are organized, and figure 3.3 shows example MusicXML file with its graphical representation. Root element of such MusicXML file is `<score-partwise>`. Its child elements we can divide into score header section and parts(one or more). Characters '?', '+' and '*' after element names are repetition operators and have the same meaning as they do in regular expressions.[2]

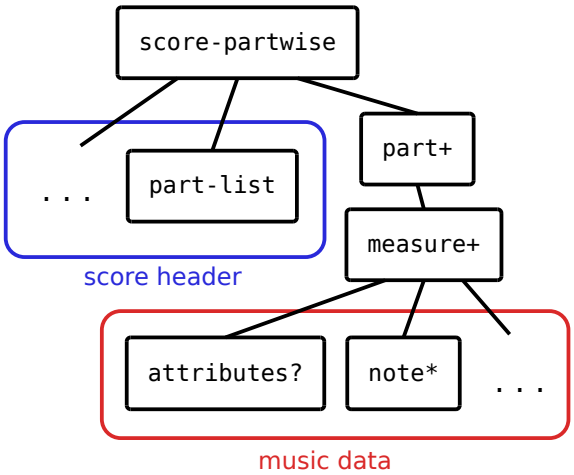


Figure 3.2: Simplified structure of MusicXML format.

[2]Number of element occurrences depending on symbol: '?' – 0-1, '+' – 1-n, '*' – 0-n.

In score header group only element, which is required, is `<part-list>` element. As its name tells, it contains information about parts in music score, such as part name, details about instrument, for which is part written, and more.

That was "organizational" section of the score, lets move into musical section. Every score must have at least one part. Each part contains one or more measures, which consists of attributes and notes. Attributes are musical informations, which can change on measure boundaries – such as clef, time signature and key signature. Note element represents musical note or rest. Since note is the most important element in music score, it deserves more detailed description.

Most important properties of note element are `pitch` and `duration`. Pitch property consists of step – name of note, `octave` and optional `alter` element – for indicating, if pitch is altered by some accidental. Duration is expressed in number telling how many divisions of quarter note particular note has in duration. `Divisions` property is defined in attributes element. Number of empty dot elements indicates number of dots of note. `Accidental` element holds name of accidental, which is shown before note. Note's `type` element contains duration of note as a string, e.g. "whole". It is optional, since duration of note can be determined from duration element.

```
<score-partwise version="3.0">
  <part-list>
    <score-part id="P1">
      <part-name>Music</part-name>
    </score-part>
  </part-list>
  <part id="P1">
    <measure number="1">
      <attributes>
        <divisions>4</divisions>
        <key>
          <fifths>3</fifths>
        </key>
        <time>
          <beats>2</beats>
          <beat-type>4</beat-type>
        </time>
        <clef>
          <sign>G</sign>
          <line>2</line>
        </clef>
      </attributes>
      <note>
        <pitch>
          <step>B</step>
          <octave>4</octave>
          <alter>-1</alter>
        </pitch>
        <duration>2</duration>
        <type>eighth</type>
        <accidental>flat</accidental>
      </note>
      <note>
        <pitch>
          <step>F</step>
          <octave>4</octave>
          <alter>1</alter>
        </pitch>
        <duration>6</duration>
        <type>quarter</type>
        <dot/>
      </note>
    </measure>
  </part>
</score-partwise>
```

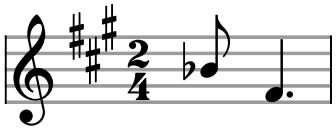


Figure 3.3: Example MusicXML file.

3.2 Research for development

In this section is described process of choosing technologies and ways how to use them for development of Web MusicXML Editor.

3.2.1 Reactive HTML5 web application

Nowadays, following technologies are being used to building common web application.

Hyper Text Markup Language (HTML) HTML is today widespread and well known language for creating web pages. It has a long history, first specifications were introduced

in the nineties of the last century. In 2014, after almost 15 years from the latest HTML version release, was published HTML5[3].

The fifth version of HTML brings many improvements. It comes with new HTML tags and simplifies syntax of some existing ones. Very powerful are new HTML5 APIs[4]. For multimedia content there are Web Audio and Media APIs, providing `<audio>` and `<video>` elements for multimedia. Graphics can be presented on web page through drawing element `<canvas>`, `<svg>` or WebGL API. For working with uploaded files there is File API, it allows to load uploaded file to program memory. Other APIs provide extensive functionality, such as Web Storage (replacement for cookies), offline work, geolocating of client, threads support through Web Workers and more.

3.2.2 Music notation rendering

After general purpose technologies, for this project is needed also specific functionality – rendering music notation online in browser.

From graphical HTML5 APIs could be used for music notation rendering Canvas and SVG. Canvas is fast and has hardware acceleration. It can draw lot of frames per second. However, it is only bitmap, and there are not any objects of rendered content. Browser only draw objects into bitmap and forgets them. This is not good for interactivity, when there are needed objects, with which can user manipulate with mouse.

Therefore, better choice is SVG. Since MusicXML and SVG are both XML documents, it would be possible to convert between them via XSLT transformation. There exists projects on such topic, e.g. ScoreSVG [2]. However, it turns out, that programming in XSLT would be harder than in JavaScript. So this option is also abandoned.

After some research right approach was found. It is VexFlow library [14], which is open-source JavaScript API for rendering music notation on web pages. It can draw notes into HTML5 Canvas and SVG. It is distributed under MIT licence.

[3]Standard of the HTML5 is described on address https://www.w3.org/TR/html5

[4]Acronym API stands for Application Programmable Interface.

Chapter 4

Existing Solutions

4.1 Open-source editors

4.1.1 LeadsheetJS

LeadsheetJS [9] [11] is JavaScript library for managing lead sheets[1]. It can be used in different configurations – as viewer, player or editor of music score.

In editor configuration – in "File" tab – user can upload MusicXML file. After upload, music notation is rendered on HTML5 `<canvas>` element using VexFlow library.

In "Notes" tab, there is possibility to select one or more notes and change their properties, such as pitch and duration. Selection can be copied and pasted. Notes can be fitted with accidentals, dots; they can be tied together, or grouped into tuplet. Notes can be also added to or removed from measure. Editor checks, if duration of notes fits the bar.

"Structure" tab provides option to transpose all piece by one to six half tones up or down, to create named sections in score and to add or remove bars. For each measure can be changed time signature, tonality, ending and label.

Music score can be exported to PNG image, PDF document or MusicCSL JSON, which is native editor format. MusicXML export is missing, so this editor cannot be fully considered as a MusicXML editor.

4.1.2 VexUI

VexUI [1] is editor based on VexFlow [14] for creating and playing music notation. It works rather like prototype of editor, since it provides only short, one row stave which user can work with. It is not possible to add new measures, only to split existing stave into measures by adding barlines.

However, way of working with notes is very comfortable. When moving mouse over stave, note is shown below cursor, so the user knows, where the note would be placed. Left mouse click simply adds note to the stave. Note is highlighted, when mouse cursor hovers above it. Clicking above or under already added note builds a chord. Right click on note opens a menu with options related to note properties.

This editor is pretty mouse oriented, middle click cycles through following score elements: notes, rests, barlines and clefs. Mouse scroll changes type of currently selected element, e.g. duration of note or rest.

[1] Lead sheet is a form of music notation that specifies only basic elements of a song – melody, lyrics and chords. It consists of only one music stave.

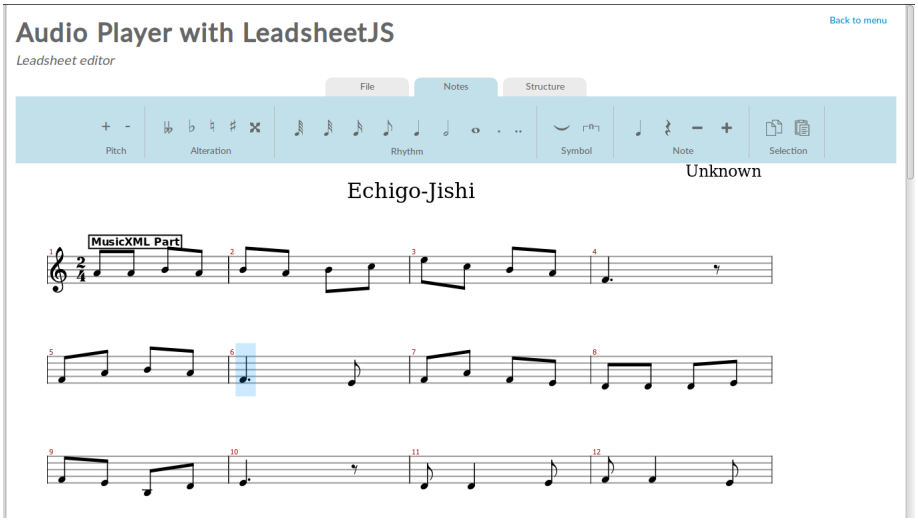


Figure 4.1: LeadsheetJS editor. [10]

As rendering layer is used HTML5 canvas element. Project allows to play created score with highlighting of currently played note. Nevertheless, it lacks file import/export.

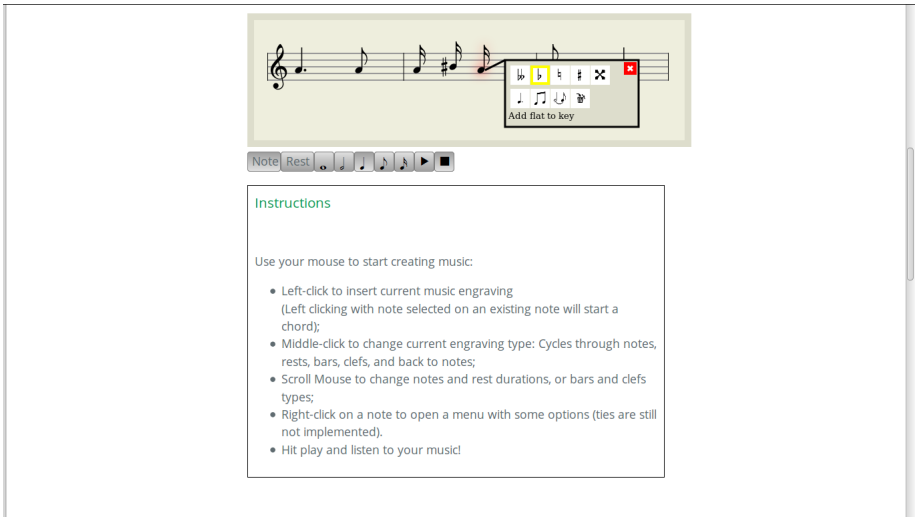


Figure 4.2: VexUI editor. [1]

4.1.3 Vexflow Notation Editor

Vexflow Notation Editor [3] is simple solution also using VexFlow [14] library and rendering on HTML5 canvas element.

User can manipulate with two music notation objects: measures and notes(or rests). Page has its own tab with set of options for these objects.

"Measures" tab offers options for adding/removing measure and setting clef, time and key signature for particular chosen measure. Measure is selected by one click, on second click can be selected particular note inside measure.

In "Notes/rests" tab user can work in two modes: selecting or adding note. When

adding, notes(or rests) are added at the end of the bar by clicking on music stave. Selected note is highlighted by square drawn around it. Then, its properties can be changed, such as value, accidental and dot. Selected note also can be removed.

Selecting voice in dropdown menu is not fully implemented, notes can be added to voice one only. As it is visible on figure 4.3, there occurs a rendering bug, where flag remains on notes, that have been beamed. File import/export is not present.

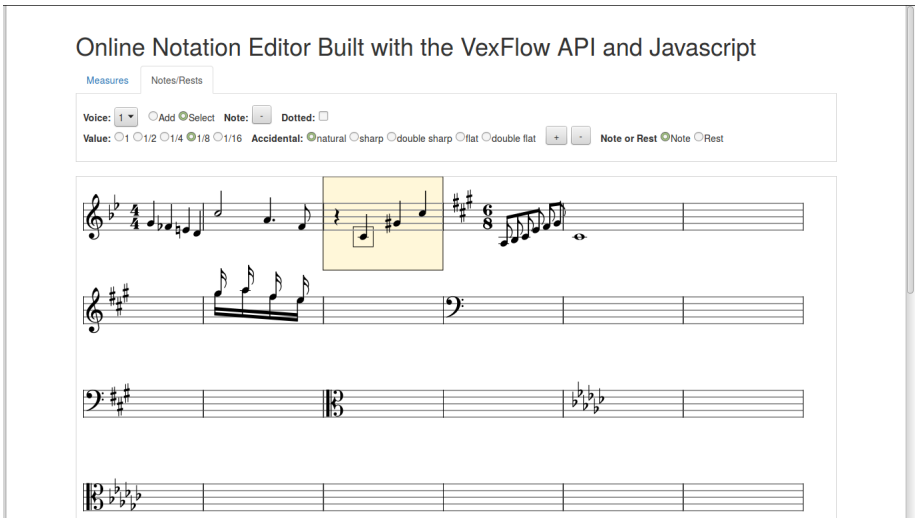


Figure 4.3: VexFlow Notation Editor. [3]

4.2 Closed-source editors

4.2.1 Noteflight

Noteligth [15] is very professional and rich-featured online music notation software. It provides dozens of options for music score editing. It supports all features from open-source projects mentioned above, so here are named only most important ones.

Notes can be manipulated either with mouse or keyboard. When note is selected or added, it is also played. Also whole score has playback function. By mouse dragging it is possible to select several measures and move them to different position in score.

Important feature for editing is undo and redo button, which are present, as well as possibility to restore previous version of edited score. As shown on the picture 4.4, piano keyboard can be displayed for viewing notes and also for adding.

On Noteflight.com every user can create account, save created music scores, and share them with other users. Users can collaborate together on music scores and add comments to them. Score can be shared also on social sites – Facebook and Twitter. Alternatively, user can send link pointing to it to someone, or even embed score on his own web page.

Application is available in Flash and in HTML5 version. The latter version renders notes into svg element. MusicXML import/export is fully supported.

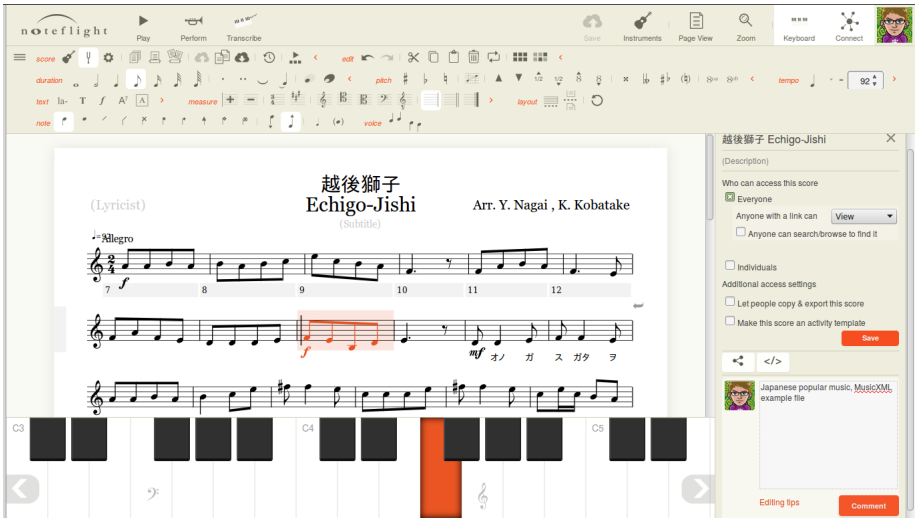


Figure 4.4: Noteflight editor. [15]

4.2.2 Flat

Flat [22] is modern and quite young online music notation application. It is not as complex as Noteflight, but its features provide enough resources for comfortable composing. Moreover, it comes with modern look.

Music notation objects, which can be added to the score are grouped into tabs. First of them is “Select” tab, where selection of bars can be copied, cut, pasted and even transposed. Selecting measures or notes is split – user can be either in measure or note selecting mode. Mode depends on chosen tab.

Another group of objects is articulation – marks added to notes, which means how note should be played. Dynamics in next tab allow add to score information regarding volume(piano, forte, etc.). Last two tabs contain lyrics and chords. Lyrics are added below music staff, whereas chords are being placed above staff.

Every object group has its own set of keyboard shortcuts. E.g. in note tab, right and left arrow keys switch between notes, up and down arrow keys change pitch of selected note, plus and minus keys add sharp or flat to note, and many more.

Flat supports user accounts, users can collaborate together on creating scores. They can view and listen to public scores of others, comment on them, or reward them with a “love button”. Score can be shared on few social sites, with link, or it can be embedded to own web page.

Notes rendering is done to svg vector graphics. Editor has feature for upload and download music score in MusicXML format.

Figure 4.5: Flat editor. [22]

4.2.3 Scorio

Scorio [18] is another commercial application for creating music scores in web browser. It is available since 2010, therefore it is not new service. It provides different concept of editing notes' pitch – and it is by mouse drag & drop operation. Notes can even be reordered by dragging one note before/after another.

Control panel provides standard editing options, such as change clef, key and time signature of measure; change duration, accidentals and articulation of note.

For drawing notes Scorio uses also different mechanism. Notes are `<div>` elements with set classes and image inside it. However, playback feature does not work.

MusicXML import/export is present, but it is paid feature.

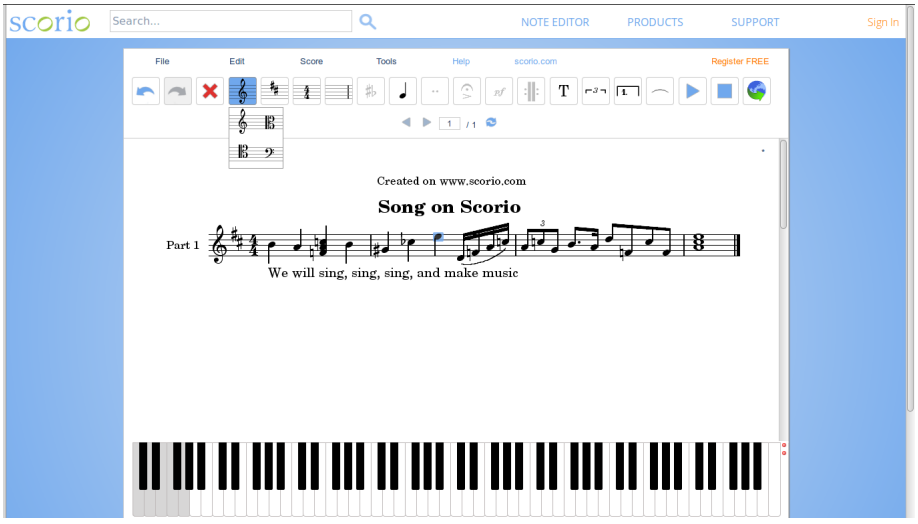


Figure 4.6: Scorio editor. [18]

4.2.4 Music Score Editor based on MusicXML

Although Music Score Editor based on MusicXML [7] is not web application, its purpose is equal to this work – to edit MusicXML files in graphical and interactive way. It is written in C# language.

4.3 Viewers and players

Apart from music score editors, there are also some solutions that provide only viewing and playing of score. Four of them, which should be mentioned are Soundslice [20], HTML5 MusicXML Viewer [13], Concerto [12] and VexFlow MusicXML plugin [16].

Chapter 5

Application design

After inspecting existing solutions, we have some idea about features, which good music notation editor should have. Let take a look at editing features also from more global perspective. There are many editors for diverse purposes. People use computers to edit digital representation of content which they value – such as text, audio, graphics, video and more. All of interactive editors share some common features, whichever their content is:

- load existing content, or make possible to create new content
- represent(visualise) content in most understandable way to user
- give user access to all parts of content via some type of cursor
- provide interface to create, change or delete content properties
- make possible to select some part of content, copy/cut it and paste it to different position
- grant opportunity to take changes back via undo/redo button
- be able to export content in used formats for specific content

5.1 Specifics of music notation editing

Music notation as content of its editor, except of general editing features requires also its specific set of features to handle notation properly. When editing music notation, editor must follow music rules, which are present in it. Editor must have some "intelligence" to fulfil following requirements:

Duration check In case to respect music theory rules, editor must check if notes fits the bar. Also it is needed to adjust notes in measure, when time signature of measure has changed.

Keep original notes' pitches after changing key signature or clef Most important is user input. Editor must respect melody, which user notated. If user changes measure attributes, which affects notes' pitch (key signature or clef), good editor should protect melody from changing.

Fluent creating of score This is feature for convenience when creating a score. When user adds last note into the last measure, editor automatically adds new measure and user can insert next notes into it without adding new measure on his own.

5.1.1 Editing operations

Editing operations, that user do, can be viewed as transactions, like in working with databases. There are 4 main requirements for database transaction: Atomicity, Consistence, Isolation, Durability.[19, p. 175 – 180] For purpose of working with music notation in this work, most important requirement is **Consistence**. It means, that music score must be valid(according to music rules) after completing particular operation on it. Therefore, operations must respect music notation rules.

Below are described operations, which user can perform in editor with musical notation objects. Some of them has to deal with respecting rules more then others.

Measures

Addition Adds measure after selected one. New measure is filled with whole rest. Measure properties, such as key signature, time signature and clef are the same as of the previous measure.

Deletion Selected measure is removed from score. If this measure had an attributes, which are the same to the following measure, they are given to that next measure.

Change of clef Clef is changed for selected measures. New clef is attached to the end of bar before selection, and on the end of last bar in selection is placed previously used clef – that is how new clef is limited only to selected measures. All notes in selected measures are transposed appropriately to new clef.

Change of key signature Key signature is changed also only for selected bars. New key signature is placed at the beginning of the first bar in selection. In first bar after selection is key signature cancelled by adding natural accidentals in place of changed key signature accidentals. Notes, which pitch is affected by new accidentals, are given a natural accidental to save their original pitch.

Change of time signature Change of time signature is also applied only for selected measures. If duration of notes exceeds the duration defined in new time signature, exceeding notes are truncated. If duration of new time signature is greater than duration of notes in bar, empty space is filled with rests. After last measure in selection the previous time signature is placed.

Notes

Addition When user selects a rest, he can place new note instead of it. Note is rendered as mouse cursor moves on stave. After click new note is added to stave. If selected object is a note, user can build a chord by adding new note above or below existing note head.

Deletion Chosen note is replaced by a rest with the same duration. If note is a part of chord, only its head is removed from chord.

Change of pitch By default, selected note is moved by one tone up or down on stave. Accidental is taken off. Some editors also have note shift by half tone or octave.

Change of duration When duration of selected note is being shortened, emerged empty place is filled with rests. With regard to prolonging, it is done only if in the same measure after selected note is enough space for it. In such case all following notes in measure are deleted and selected note is prolonged.

Addition of accidental Each accidental(double flat, flat, natural, sharp, double sharp) has its own button, which toggles displaying of such accidental. Note can have only one accidental at one time.

Addition of dot Presence of dot for selected note is toggled after checking dot checkbox. For this operation applies same rules according to changing length of note as for duration change operation.

5.2 Design

After specifying requirements on editor, designing of application can begin.

5.2.1 Base project

As a base for my work I used VexFlow Notation Editor [3] released under MIT license. This project helped me to understand how music notation editor based on VexFlow could work. However, the project had a several bugs[1] and was incomplete. I took it as a starting point for my work. Unlike the core and functionality, basic graphical user interface elements remained similar, but was extended.

5.2.2 User interface

The user interface of the Application consists of two areas. First of them is control panel, which in three tabs provides access to all functions of editor. Drawing area is the second, it is where music score is rendered. I will refer it as canvas[2], because notes are drawn here.

Control panel Panel with feature buttons and select boxes is fixed to the top of the page. When music score is long, and user must scroll down, panel will stay at the top independently on scrolling. The panel consists of three tabs with specific content for each of them. Their description follows.

[1] Bug in term of error in computer program

[2] Not to be confused with HTML5 element `<canvas>`, used element for drawing is `<svg>`.

File The File tab offers loading and saving music score from/to MusicXML file. Select drop-down list on left side allows to load an example file from directory tree of project. Upload is done with AJAX technology using jQuery library. Via file input in the middle user can upload his own file from local disc. Music score is rendered automatically after upload in both cases. After click on the button on the right browser offers download of music score in a MusicXML format. File tab is added to program interface, the base project did not have file support.

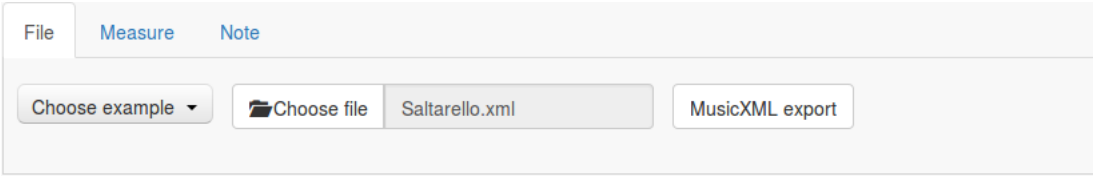


Figure 5.1: Content of file tab.

On this place it is important to note, that editor works in two modes. First one is measure mode, when user manipulates with measures. The second is note mode – working with notes. In either mode there is always one object in selection – measure or note. All operations of mode are performed on that selected object.

Measure Description from the left: Buttons with + and - signs add or remove measure. Selecting clef in its dropdown menu assigns this clef to the selected measure. Key signature select box works in the same manner. Controls for time signature consists of two select dropdowns and one button. After clicking on button time signature with selected numbers in select boxes are added to selected bar. When user clicks on some measure, values of all select boxes are automatically changed according to attributes of selected measure.

Figure 5.2: Content of measure tab.

Note Note object has generally two basic properties: pitch and duration. Dot element belongs to duration property and accidental to pitch. Pitch operations can be performed only on notes, since rests do not have a pitch. Pitch editing buttons remove accidental from selected note(if any) and push note one position down or up on the stave.

Icons of notes are used to change note's or rest's duration. Only one duration can be selected at one time. Dot checkbox toggles note's or rest's dot. Accidental buttons toggle or replace accidental of note. Delete button turns note into a rest with the same duration.

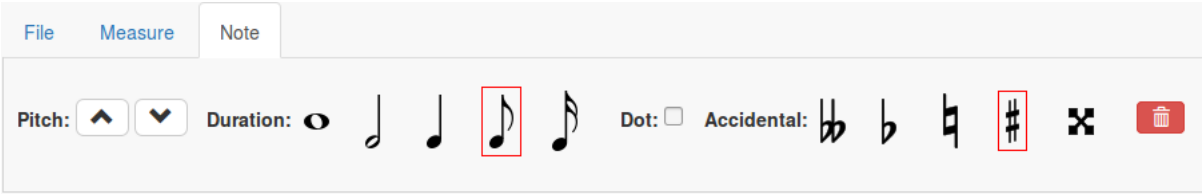


Figure 5.3: Content of note tab.

Canvas As *canvas* it is meant area for drawing music score. Actually, element responsible for rendering is `<svg>`. This HTML5 element is wrapped by some `<div>` elements with Bootstrap classes for responsivity – more details are provided in following Implementation chapter. Therefore, in this thesis *canvas* is term used for `<svg>` wrapped by `<div>` elements.

Rendered music score responds on window resize(includes also zooming) and it redraws itself on such events. Also when music score is long, *canvas* is extended so it can display whole music piece and scroll bar is shown.

Musical content in SVG is reactive. Measures and notes get highlighted when mouse cursor is over them. By left mouse click particular measure or note is selected and highlighted with different colour.

On figure 5.4 is shown look of whole application, both with control panel and *canvas*.

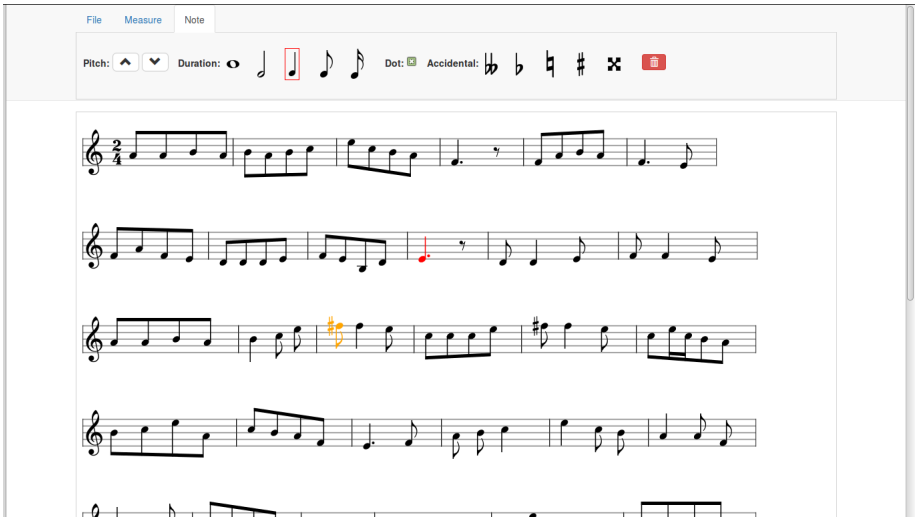


Figure 5.4: Appearance of Web MusicXML Editor application.

Chapter 6

Implementation

6.1 Used technologies

In following sections are described major web technologies, which were studied in chapter 3, and ways how I used them in my project.

6.1.1 HTML5

Since Web MusicXML Editor is currently single page application, only one HTML file is used in this project, which is `index.html`. In `<head>` element there are title of page and links to used style sheets. Page `<body>` contains control panel (contained in HTML5 `<nav>` element) consisting of three tabs and their content. For music score rendering area is used `<svg>` element, wrapped by `<div>` elements with set classes for Bootstrap functionality. In page `<footer>` there are `<script>` tags for loading remote and local libraries, project source scripts and small inline script. For uploading and reading content of file is used HTML5 `FileReader API`.

6.1.2 CSS

There are two ways in which cascading style sheets are used in this project. The first are external definitions in separate file `style.css`. To avoid overlapping fixed to top control panel and body of page, `padding-top` attribute for body element is used. Further styles definitions pertains to positioning of tools on control panel, borders around elements and size of icons.

The second usage of CSS are inline styles, which are part of HTML elements in their `style` attribute. Their usage should be kept at minimum, as they violates purpose of CSS – to separate content from presentation. In my work two inline styles are used for minor positioning adjustment.

6.1.3 JavaScript

JavaScript is the major language in which is this application written. All source files are linked in footer of HTML file via `<script>` tags. Inside the last one is inline script, it initializes the program by calling three specific functions. Link(`<a>`) and `<button>` HTML elements have `onclick` attribute, in which are calls to JavaScript functions.

6.1.4 jQuery

jQuery library provides many useful utilities for JavaScript programming. In this project it is used for several tasks. Most often way of usage is selecting elements in DOM tree of HTML and even SVG document. With selected elements are performed actions such as adding event listeners, changing style or removing. For completing one particular operation I added my custom function to jQuery functions, so that I can directly call it on elements. To asynchronously upload example file I used `ajax` function.

For better look and enhanced functionality of HTML select boxes I employed jQuery plugin called SelectBoxIt [4]. When its initialization method is invoked on HTML `<select>` element, object is returned with its own available methods. Further work with select box is through this object. It requires jQueryUI [5], as a dependency, which is also included. For select boxes I selected Bootstrap style to make look of application consistent.

6.1.5 Bootstrap

Framework Bootstrap I use in CSS and Javascript version. Most important usage is for fixing control panel to the top of the page. This functionality is done by setting class `navbar-fixed-top` to control panel `<nav>` element.

Switching between tabs is also done by Bootstrap. Tabs are list items with links in unordered list, and tab contents are div elements. With appropriate classes and attributes in elements is tab switching put into work, and even with fade in effect. SVG for drawing notes is wrapped in `<div>` elements with classes `container`, `row` and `col-xs-12`, which provides responsivity for all screen sizes. Functionality described in this paragraph was already part of base project, which is mentioned in section 5.2.1.

Bootstrap also comes with pretty style for buttons. I use default style for normal buttons, and `btn-danger`(red) style for button for deleting note. Icons on buttons come from Glyphicon Halflings set[1] and are available through Bootstrap.

With regard to HTML file input, I made it to look better using plugin Bootstrap FileStyle [8].

6.1.6 Open-source projects

For the most important task – notes rendering – is used VexFlow [14], which is open-source JavaScript API for rendering music notation on web pages. It can draw notes into HTML5 Canvas and SVG. It is distributed under MIT licence.

Few conversion functions and tables between MusicXML and VexFlow structures are taken from two files of Concerto [12] project. Inspiration for extending original VexFlow objects with my own methods comes from `VexFlowExtension.js` file of VexUI [1] project. Both Concerto and VexUI are released under MIT licence.

For conversions between XML and JSON I use two scripts written by Stefan Goessner – `xml2json.js` and `json2xml.js`, licensed under LGPL 2.1 licence. These conversion are in my project necessary for file import/export.

[1] Home page on http://glyphicons.com

6.2 Application architecture

Application source code is split into "modules". Each module represents operation, which can be done with musical objects. Inside particular module there are functions named by subjects of module operation. Every function performs module operation on its subject.

Most of the module functions do not have arguments. Such functions operates with selected measure or note, which id is stored in `editor.selected` object. Id is used for indexing global array of measures or notes to get selected measure or note, with which function can do its work.

```
editor = {

    add: {          delete: {      draw: {            edit: {           parse: {
     accidental(),   accidental(),  measure(),         noteAccidental(), all(),
     clef(),         measure(),     selectedMeasure(), noteDot(),        attributes(),
     keySignature(), note()         score()            noteDuration(),   measure(),
     measure(),     },             },                  notePitch()       note()
     note()                                           },                }
    },

};
```

Figure 6.1: Code modules.

6.2.1 Data structures and flow

This section describes representation of musical data in application.

Music score models Music score in program is represented by 3 data models, each with its own purpose. Model's descriptions follows.

JavaScript Object Notation (JSON) Variable `scoreJson` holds JSON object describing whole music score. This JSON represents uploaded or created MusicXML file – in fact, it is converted version of it. Every change performed in editor is reflected to `scoreJson`. When exporting, this variable is taken, converted to XML format and given to user for download.

VexFlow objects Rendering library VexFlow is using its own structures for representing music score. When file is uploaded and converted into JSON, `editor.parse.all()` function parses JSON into VexFlow objects. For measure the type of VexFlow object is `Vex.Flow.Stave` and for note `Vex.Flow.StaveNote`. These objects are stored in global arrays available for drawing through `editor.draw.score()` function.

Scalable Vector Graphics (SVG) SVG is graphical representation of music piece. It is result of VexFlow library job, which creates it and inserts it into another `<svg>` element (which serves as a wrapper) in `index.html`.

All direct descendants of root `<svg>` element are svg groups elements (`<g>`), which encase measures. These groups for measures have set id containing measure index, and class `vf-stave`. Inside measure group are graphical elements such as staff and bar lines, glyphs for clef, time and/or key signature; also rectangle with event listeners (mouse over

& out, mouse click) for highlighting and selecting measure; and notes, each wrapped in its own group.

Note group element has id consisting of measure and note id, and class named `vf-stavenote`. Inside element there are two other groups: one with graphical parts of note, namely note head, stem, flag; and second with note modifiers (accidentals). Event listeners are also added – for highlighting note, when hovered by mouse; and for selecting note on mouse click.

Table 6.1 shows how in every music score model note object can be accessed. Highlighted characters are indices of measure (m) and note inside it (n).

model	access to note object
JSON	`scoreJson["score-partwise"].part[0].measure[m].note[n]`
VexFlow	`gl_VfStaveNotes[m][n]`
SVG	`jQuery("svg > g > g#vf-mmnn.vf-stavenote")`

Table 6.1: Accessing note object in different music score models.

MusicXML File Data Flow

As this application is editor of MusicXML files, program works a lot with MusicXML data. Therefore, it is important to work with them efficiently and quickly. For better parsing and manipulation with data, they are converted into native notation of JavaScript objects – JSON. Figure 6.2 shows how MusicXML data flows through application.

After upload via file input, `FileReader` reads MusicXML file into string. This string is parsed into `XML Document` with `jQuery.parseXML()` function. Such document is converted to JSON string by `xml2json()`. From the string is created JavaScript object by `jQuery.parseJSON()`.

With the object, which is stored in variable `scoreJson`, editor further works during user editing session.

When user requests MusicXML export, `scoreJson` object is given to `json2xml()` function and is converted into XML string. The string is encoded with `encodeURIComponent()`, joined with XML header and put into `href` attribute of link. After that, browser offers user to download MusicXML file.

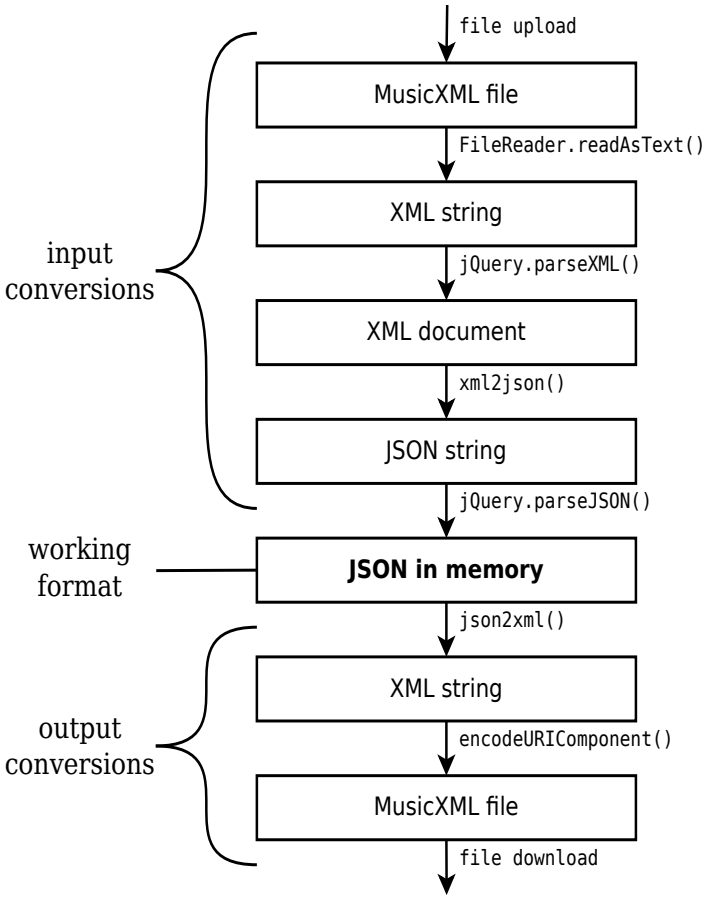


Figure 6.2: MusicXML dataflow through program.

Chapter 7

Testing and Evaluation

When developing online application, programmer does not have to care about computer architecture or operating system. However, he does have to be concerned about web browser support. Behaviour and JavaScript implementation of various browsers differs. This application was tested in several browsers. Following paragraphs describes results:

Mozilla Firefox 46.0 Firefox is the main browser, whole development was done in this browser. It is recommended to use application in Mozilla Firefox. No browser problems appear with this browser, only possible issues are on application side, which are known and will be fixed.

Google Chrome 50.0 & Opera 37.0 These two browsers are based on the same rendering core WebKit, so application behaves in both browsers similarly. Two issues are present. Dot element of notes is for unknown reason too large, that it overlaps another measures, which becomes unreachable for selecting together with their notes. For fixing this issue some research must be done to determine, what causes this behaviour. Second problem is minor, input controls in File tab are displaced a bit. It should be corrected by css positioning attributes. Everything else works as expected.

Internet Explorer 11.0 In Internet Explorer manipulating with notes works correctly. Results from testing are quite good, since this browser is known to behave more differently from the others. However, one significant feature does not work – file export. It could be fixed by using `Blob` type for file in JavaScript and/or another mechanism for downloading file.

MusicXML parsing was tested on example set of files from MusicXML web page[1]. Most of the files have been imported without problems. However, with some of them (cca 2 of 12) occured error during parsing due to non standard note duration. This issue should be fixed by reading also `<type>` element of the note, if the duration is not correct.

Web Editor of MusicXML Files is valid HTML5 application, its code has successfully passed through W3C Validator.

[1] http://www.musicxml.com/music-in-musicxml/example-set/

Chapter 8

Conclusion

The task of this thesis was to implement web application for editing files in MusicXML format.

8.1 Achieved goals

Editing files of MusicXML format MusicXML file can be uploaded, edited and downloaded back to user. Although editor does not support all MusicXML elements, it respects them and stays them untouched – it does not delete or modify them.

Notes rendering Application renders note in standard graphical notation. Due to vector nature of SVG, music score is at high quality at every level of zooming.

Graphical note designer Music score is interactive and enables user to work directly with notes via mouse cursor. Measures and notes reacts on user mouse events and are being highlighted when hovered or clicked.

Personal learning Important benefits for me personally were studying structure of MusicXML format and learning to work with technologies for notes rendering and editing. I learned also new things about music notation. It is sure, that this bachelor thesis was for me really helpful preparation for future work.

8.2 Shortcomings and possible improvements

Duration check The biggest limitation is currently possibility to add only one note to measure. Editor does not have implemented mechanism for check if notes fits in the measure. This feature must be implemented to make editor really usable.

Multi part score Currently editor supports only music scores containing one music part. However, even with multi-part score it renders first part and makes it editable. Implementation, for which program is ready, would be to make possible to switch between parts and modify them separately. For rendering multiple parts it would take more effort to implement it.

Sophisticated intelligence of editor Application supports some level of "intelligence", as described in 5.1, but surely it should be extended.

8.3 Future steps

The application is published as open-source under MIT license and is available on GitHub[1]. Following step is to build a community around the project and make it grow.

Planned extensions are to support more features from MusicXML 3.0 specification, such as chords, multi-part music score and more. For even better interactive experience it is possible to implement interacting with notes via keyboard, or mouse dragging. Also important is music playback and export to more formats such as SVG and PDF. And the biggest goal is to make it a social site for music score sharing.

[1] https://github.com/freetomik/web-musicxml-editor

Bibliography

[1] André Bakker. VexUI – A VexFlow Editor for creating and playing music. [online]. URL http://andrebakker.github.io/VexUI, [cited 2016-04-17].

[2] Geoffrey Alan Bays. ScoreSVG – A New Software Framework for Capturing the Semantic Meaning and Graphical Representation of Musical Scores Using JAVA2D, XML, and SVG. Thesis, Georgia State University, Atlanta, 2005-08-08.

[3] Myles English. Vexflow Notation Editor – Online Notation Editor Built with the VexFlow API and Javascript [online].
URL https://github.com/Bijingus/vexflow-notation-editor, [cited 2016-04-17].

[4] G. Franko. SelectBoxIt. [online].
URL http://gregfranko.com/jquery.selectBoxIt.js, [cited 2016-05-18].

[5] D. Furfero. jQuery UI Touch Punch. [online].
URL http://touchpunch.furf.com, [cited 2016-05-18].

[6] Michael Good. Representing music using xml. In *ISMIR*, 2000.

[7] Najeeb Ullah Khan and Jung-Chul Lee. Development of a Music Score Editor based on MusicXML. *Journal of the Korea Society of Computer and Information*, vol. 19(issue 2):77–90, 2014-02-28.

[8] Markus Lima. Bootstrap FileStyle. [online].
URL http://markusslima.github.io/bootstrap-filestyle, [cited 2016-05-18].

[9] Daniel Martín and Timotée Neullas in SonyCSL Paris Music Team. LeadsheetJS – A Javascript Library for Online Lead Sheet Editing [online].
URL http://lsdb.flow-machines.com/leadsheetJS, [cited 2016-04-18].

[10] Daniel Martín and Timotée Neullas in SonyCSL Paris Music Team. LeadsheetJS editor [online].
URL http://lsdb.flow-machines.com/leadsheetJS/example_editor_menu, [cited 2016-04-18].

[11] Daniel Martín, Timotée Neullas, and François Pachet. LEADSHEETJS: A Javascript Library for Online Lead Sheet Edition. In M. Battier, J. Bresson, P. Couprie, C. Davy-Rigaux, D. Fober, Y. Geslin, H. Genevois, F. Picard, and A. Tacaille, editors, *Proceedings of the First International Conference on Technologies for Music Notation and Representation - TENOR2015*, pages 1–10, Paris, France, 2015. Institut de Recherche en Musicologie.

[12] Taehoon Moon. Concerto Project – A Javascript library for rendering MusicXML using MusicJSON and VexFlow. [online].
URL https://github.com/panarch/concerto, [cited 2016-04-19].

[13] Musicxml-viewer.com. HTML5 MusicXML viewer [online].
URL
http://www.musicxml-viewer.com/features/basic-usage/basic-usage.html,
[cited 2016-04-18].

[14] Mohit Muthanna. VexFlow – Music Engraving in JavaScript and HTML5 [online].
URL https://www.vexflow.com, [cited 2016-04-18].

[15] Noteflight. Noteflight – Online Music Notation Software [online].
URL https://www.noteflight.com, [cited 2016-04-18].

[16] Dan Ringwalt. VexFlow MusicXML plugin – A plugin for parsing and engraving MusicXML documents in VexFlow. [online].
URL https://github.com/ringw/vexflow, [cited 2016-05-04].

[17] Baron Schwartz. Transforming XML Into Music Notation. Bachelor thesis, University of Virginia, The Faculty of the School of Engineering and Applied Science, Charlottesville, 2003-04-10.

[18] Scorio. Scorio.com – Scorio Online Editor [online].
URL https://www.scorio.com, [cited 2016-05-11].

[19] Abraham Silberschatz, Henry F Korth, Shashank Sudarshan, et al. *Database system concepts*. McGraw-Hill New York, 4 edition, 2001.

[20] Soundslice. Free MusicXML viewer [online].
URL https://www.soundslice.com/musicxml-viewer, [cited 2016-04-18].

[21] Hope Strayer. From neumes to notes. *Musical Offerings*, vol. 4(issue 1):1–14, 2013.

[22] Tutteo. Flat.io – The online and collaborative music notation software [online].
URL https://www.flat.io, [cited 2016-04-18].

Appendices

List of Appendices

Appendix A

Content of the CD

The CD attached to this thesis contains following directories and files:

Directory source

- Directory containing all source files needed for running web application.

Directory thesis

- Directory with LaTeX source files of thesis.

Directory examples

- Directory with example MusicXML files.

File bachelor_thesis.pdf

- Technical report in PDF format.

Appendix B

Manual

Following steps to run application:

1. Navigate into `source` directory.
2. Open `index.html` file.
3. Application will be opened in the web browser.