Testing Document

February 26, 2021

Tabs2XML

Alp Sirek
Andrew Ngov
Arjit Johar
Daniel Santorelli
Muhammad Azizi

Submitted in Fulfillment of The Final of
EECS 2311 Software Development Project

Table of Contents

Introduction

Tabs2XML is developed for the purpose of converting guitar tablature files and drums tablature files into MusicXML files. Due to the relatively new format, there aren't many music pieces written in MusicXML. While tablature for guitars and drums are easy to understand, they offer a low degree of readability and modification. The MusicXML format builds on these shortcomings to allow readers to better understand the music piece and easily play it. Tab2xml is developed for those who want to play songs in the format of MusicXML, but find that they can only find the tablature of those pieces (common occurrence as there aren't as many pieces of song in the MusicXML format). Tabs2XML also allows users to freely modify their music pieces. Tabs2XML is also developed for those who only have a tablatures for a song but want to view it in a music sheet. Tabs2XML converts tablatures into MusicXML files that can be modified and viewed as a music sheet using a third party app, such as MuseScore.

Objectives

This document is to test the functionality of the system and make sure the program can handle different cases and errors properly. The test is to make sure that the program behaves as expected for the users.

Guitar Test Cases

The expected output files for cases 1-7 are included in the GitHub repository for reference.

Test Case 1: Course Wiki

```
|-----------0-----|-0---------------|
|---------0---0---|-0---------------|
|-------1-------1-|-1---------------|
|-----2-----------|-2---------------|
|---2-------------|-2---------------|
|-0---------------|-0---------------|
```

This is the example tablature text provided by the EECS2311 wiki. It is the base test case that several others are based on. This test is to ensure that the program is functional and can convert basic tablatures.

Test Case 2: Edited Numbers

```
|-----------1-----|-2---------------|
|---------1---2---|-1---------------|
|-------2-------0-|-0---------------|
|-----0-----------|-1---------------|
|---0-------------|-0---------------|
|-1---------------|-2---------------|
```

This case is based on Test Case 1, but with edited number values. Meant to test how the program handles differing number values. This test is to see if the program properly converts edited tablatures from the one given in the EECS2311 wiki.

Test Case 3: Edited Heights

```
|-1---------------|-2---------------|
|---0-------1-2---|-1---------------|
|-----------------|-0---------------|
|-----0---1-------|-1---------------|
|-----------------|-0---------------|
|--------2------0-|-2---------------|
```

This case is based on Test Case 1 and 2. In this case, the numbers are shifted around vertically. This is used to test how the program handles numbers on different lines.

Test Case 4: Empty

```
|-----------------|-----------------|
|-----------------|-----------------|
|-----------------|-----------------|
|-----------------|-----------------|
|-----------------|-----------------|
|-----------------|-----------------|
```

This case is meant to test how the program handles an empty tab file.

Test Case 5: Incorrect Format

```
|--------|-----------------|
|-----------------|-----------------|
|-------|-----------------|-----------------|
|-----------------|-----------------|
text should result in an error
|-----------------|-----------------|
```

This case is meant to test how the program handles a tab file of incorrect formatting. These formatting errors include incorrect amounts of '-' characters and non-tablature text in one of the lines.

Test Case 6: Not Enough Lines

```
|-----------0-----|-0---------------|
|---------0---0---|-0---------------|
|-------1-------1-|-1---------------|
|-----2-----------|-2---------------|
|---2-------------|-2---------------|
```

This case is based on Test Case 1, but with the bottom line removed. Meant to test how the program handles the input error of missing a line.

Test Case 7: Too Many Lines

```
|-----------0-----|-0---------------|
|---------0---0---|-0---------------|
|-------1-------1-|-1---------------|
|-----2-----------|-2---------------|
|---2-------------|-2---------------|
|-0---------------|-0---------------|
|-0---------------|-0---------------|
```

This case is based on Test Case 1, but with an extra bottom line added. Meant to test how the program handles the input error of having too many lines.

Test Case 8: Grace notes

```
|-----------0-----|-0---------------|
|---------0---0---|-0---------------|
|-------g0h1----1-|-1---------------|
|-----2-----------|-2---------------|
|---2-------------|-2---------------|
|-0---------------|-0---------------|
```

This test case is an example of a grace note provided by the EECS2311 wiki and it is to test if the program can handle and properly convert more advanced tablatures that include grace notes.

Test Case 9: Repeated Measures

```
|-----------0-----||----------0--------4|
|---------0---0---||----------0--------||
|-------1-------1-||*---------1-------*||
|-----2-----------||*---------2-------*||
|---2-------------||------2---2--------||
|-0---------------||--0-------0--------||
```

This test case is an example of repeated measures in a tablature provided by the EECS2311 wiki and it is to test how the program handles repeated measures.

Why These Test Cases (Guitar)?

All of the cases above, when combined, provide sufficient testing for the program from every angle of use. Test Case 1 serves as the most basic test case, while the subsequent cases serve to test the program under different conditions. Cases 1-4 test the program's functionality under expected conditions, while cases 4-7 test the program's robustness in several different error scenarios (they test the program's ability to spot erroneous inputs and notify the user of their specific mistake). Cases 8-9 tests to see if the program is able to handle more advanced tablatures that include grace notes, repeated measures, hammer-ons and pull-offs, etc. The test cases make sure that errors do not crash the program and inform the user what the error is. Also, it tests the features that were required for the EECS2311 project to see if the program behaves as expected and can convert them.

Drum Test Cases

Test Case 1: Course Wiki

```
CC|x---------------|--------x-------|
HH|--x-x-x-x-x-x-x-|----------------|
SD|----o-------o---|oooo------------|
HT|----------------|----oo----------|
MT|----------------|------oo--------|
BD|o-------o-------|o-------o-------|
```

This is the example tablature text provided by the EECS2311 wiki. It is the base test case that several others are based on. This test is to ensure that the program is functional and can convert basic drum tablatures.

Test Case 2: Edited Tablature

```
CC|x------x-x--x---|--------x-------|
HH|----x-x---x-x---|-----x---x--x---|
SD|----o-------o---|---------o-o----|
HT|-----o-----o--o-|----oo----------|
MT|-------o----o---|------oo--------|
BD|o-------o-------|o-------o-------|
```

This case is based on Test Case 1. This tests the program and how it can handle different tablatures and if it behaves as expected.

Test Case 3: Empty

```
|----------------|----------------|
|----------------|----------------|
|----------------|----------------|
|----------------|----------------|
|----------------|----------------|
|----------------|----------------|
```

This case is meant to test how the program handles an empty tab file. This is to test how the program deals with an empty tablature and if the conversion works properly.

Test Case 4: Incorrect Format

```
|--------|------x----------|
|----------x------|-----------------|
|-------|-o----o----------|-----------------|
|-------x---------|-----------------|
text should result in an error
|-----------------|-----------------|
```

This case is meant to test how the program handles a tab file of incorrect formatting. These formatting errors include incorrect amounts of ‘-’ characters and non-tablature text in one of the lines. This is to test if the program can properly handle incorrect tablature and if it returns an error.

Test Case 5: Not Enough Lines

```
CC|x---------------|--------x-------|
HH|--x-x-x-x-x-x-x-|----------------|
SD|----o-------o---|oooo------------|
HT|----------------|----oo----------|
MT|----------------|------oo--------|
```

This case is based on Test Case 1, but with the bottom line removed. Meant to test how the program handles the input error of missing a line. This is to test if the program can recognize the missing measure and report the error to the user.

Test Case 6: Too Many Lines

```
CC|x---------------|--------x-------|
HH|--x-x-x-x-x-x-x-|----------------|
SD|----o-------o---|oooo------------|
HT|----------------|----oo----------|
MT|----------------|------oo--------|
BD|o-------o-------|o-------o-------|
BD|o-------o-------|o-------o-------|
```

This case is based on Test Case 1, but with the bottom line shown twice. Meant to test how the program handles extra lines. This is to test if the program can recognize the incorrect tablature formatting and display an error.

Why These Test Cases (Drums)?

All of the cases above, when combined, provide sufficient testing for the program from every angle of use. Test Case 1 serves as the most basic test case, while the subsequent cases serve to test the program under different conditions. Cases 1-3 tests the program's functionality for basic tablature and if it can handle and convert them properly. Cases 4-6 test how the program handles incorrect formatting of tablatures (or non-tablature inputs) and its robustness in error detections (test the program's ability to detect error in the input file and display an error message). The test cases test the features that were required for the EECS2311 project (convert simple drum tablatures and detect incorrect inputs) to see if the program behaves as expected and can convert them.

Bass Test Cases

Test Case 1: Basic Tablature

```
G|-----------0-----|-0---------------|
D|---------0---0---|-0---------------|
A|-------1-------1-|-1---------------|
E|-----2-----------|-2---------------|
```

This is an example of a very basic bass tablature. It is the base test case that several others are based on. This test is to ensure that the program is functional and can convert basic tablatures.

Test Case 2: Edited Numbers

```
G|-----------1-----|-2---------------|
D|---------1---2---|-1---------------|
A|-------2-------0-|-0---------------|
E|-----0-----------|-1---------------|
```

This case is based on Test Case 1, but with edited number values. Meant to test how the program handles differing number values. This test is to see if the program properly converts edited tablatures from the one given in the EECS2311 wiki.

Test Case 3: Edited Heights

```
G|-1---------------|-2---------------|
D|---0-------1-2---|-1---------------|
A|-----------------|-0---------------|
E|-----0---1-------|-1---------------|
```

This case is based on Test Case 1 and 2. In this case, the numbers are shifted around vertically. This is used to test how the program handles numbers on different lines.

Test Case 4: Empty

```
|-----------------|-----------------|
|-----------------|-----------------|
|-----------------|-----------------|
|-----------------|-----------------|
```

This case is meant to test how the program handles an empty tab file.

Test Case 5: Incorrect Format

```
|--------|-----------------|
|-----------------|-----------------|
text should result in an error
|-----------------|-----------------|
```

This case is meant to test how the program handles a tab file of incorrect formatting. These formatting errors include incorrect amounts of '-' characters and non-tablature text in one of the lines.

Test Case 6: Not Enough Lines

```
|-----------0-----|-0---------------|
|---------0---0---|-0---------------|
```

This case is based on Test Case 1, but with the bottom line removed. Meant to test how the program handles the input error of missing a line.

Test Case 7: Too Many Lines

```
|-----------0-----|-0---------------|
|---------0---0---|-0---------------|
|-------1-------1-|-1---------------|
|-----2-----------|-2---------------|
|---2-------------|-2---------------|
|-0---------------|-0---------------|
|-0---------------|-0---------------|
```

This case is based on Test Case 1, but with an extra bottom line added. Meant to test how the program handles the input error of having too many lines.

Test Case 8: Grace notes

```
G|-----------0-----|-0---------------|
D|---------0---0---|-0---------------|
A|-------g0h1----1-|-1---------------|
E|-----2-----------|-2---------------|
```

This test case is an example of a grace note provided by the EECS2311 wiki and it is to test if the program can handle and properly convert more advanced tablatures that include grace notes.

Test Case 9: Repeated Measures

```
G|-----------0-----||----------0--------4|
D|---------0---0---||----------0--------||
A|-------1-------1-||*---------1-------*||
E|-----2-----------||*---------2-------*||
```

This test case is an example of repeated measures in a tablature provided by the EECS2311 wiki and it is to test how the program handles repeated measures.

Why These Test Cases (Bass)?

All of the cases above, when combined, provide sufficient testing for the program from every angle of use. Test Case 1 serves as the most basic test case, while the subsequent cases serve to test the program under different conditions. Cases 1-4 test the program's functionality under expected conditions, while cases 4-7 test the program's robustness in several different error scenarios (they test the program's ability to spot erroneous inputs and notify the user of their specific mistake). Cases 8-9 tests to see if the program is able to handle more advanced tablatures that include grace notes, repeated measures, hammer-ons and pull-offs, etc. The test cases make sure that errors do not crash the program and inform the user what the error is. Also, it tests the features that were required for the EECS2311 project to see if the program behaves as expected and can convert them.