Joshua Bonn

bonn7152@vandals.uidaho.edu

https://github.com/JoshuaBonn1

IRC nick: jbonn

Phone- (503)-991-3668

3890 Pleasant View Dr. NE Keizer, OR 97303

Synopsis

MuseScore is a free, open-source score writer focused on creating, playing, and printing beautiful music. MuseScore has many features to make editing scores of any size easy and intuitive. The current mode of large score navigation is a tool called Navigator. It displays a small version of the visual score that you drag a small box through to change the current visual focus. It is tough to visualize where you are in a large score with a minimized score. That is where the timeline idea comes in.

The timeline would display an abstract view of the score, distilling it down to measure numbers, key signature, time signature, and tempo changes, as well as a list of the instruments. These would be displayed in a grid format. The user can select a spot on the grid and be automatically placed there in the scroll. Many features would exist to help the user navigate the timeline and understand the abstraction.

Benefits to MuseScore

The timeline would benefit the creators on MuseScore to have a quick, comprehensive method to navigate large scores. The current system works well for small pieces, but not as well for large works. It would also aid in the understanding of large works by being a visual assistant of a simple abstraction of the score. The timeline would work smoothly with both large and small works.

Another benefit would be the addition of instrument categorization. This could be used for many different purposes such as bring instruments of the same type closer together in the typeset or having an automatic reorganizer of instruments.

Deliverables

- Integration of the timeline into MuseScore as a tool.

- A complete framework for instrument categorization.
- Full documentation for the above deliverables.

Project Details

There are two main categories to the timeline: grid layout and interaction with the timeline.

Grid Layout

The grid layout is like the picture below. As shown, the grid would show measure numbers, tempo changes, key signature changes, time signature changes, and all the instruments in the score. Other features include: rehearsal numbers or letters, if they are in the score; separation of movements and; tabs at the top of the navigator for movement numbers and tempo changes, described further in the interaction section.

The light blue coloring represents measures with notes in them. There would be a variance in saturation depending on the density of notes in the measure. This density function depends on the number of beats in the measure to the number of notes. The current box that is being worked on would be highlighted with a thick border as well as a column box to indicate the current measure being addressed.

Parts of the timeline would be collapsible to save space and allow for users to see only the important parts. The collapsible sections include: all the instruments, so only the measure numbers, tempo, key signature, and time signature changes would display; same instruments, such as Flute and Flute II becomes just Flute Group; similar instruments, such as the different saxophones; instrument classes, like woodwinds, brass, strings, etc.; and user defined sections, the method of selection is described in the interaction section. Collapsing the parts would lead to a logical OR of the notes between all the parts. The functionality of part separation would lead to the creation of a general part categorizing system.

At the top of the timeline area, there would be tabs that present the movement numbers. When they are interacted with, defined later, a group of sub tabs appear with a list of the tempo changes.

Interaction

This section describes the interaction between the user and the timeline to change the location on the main scroll. The main interaction for the timeline is clicking a box and automatically being moved to its

related position in the scroll. This interaction also applies to tempo changes, key signature changes, time signature changes, and rehearsal numbers.

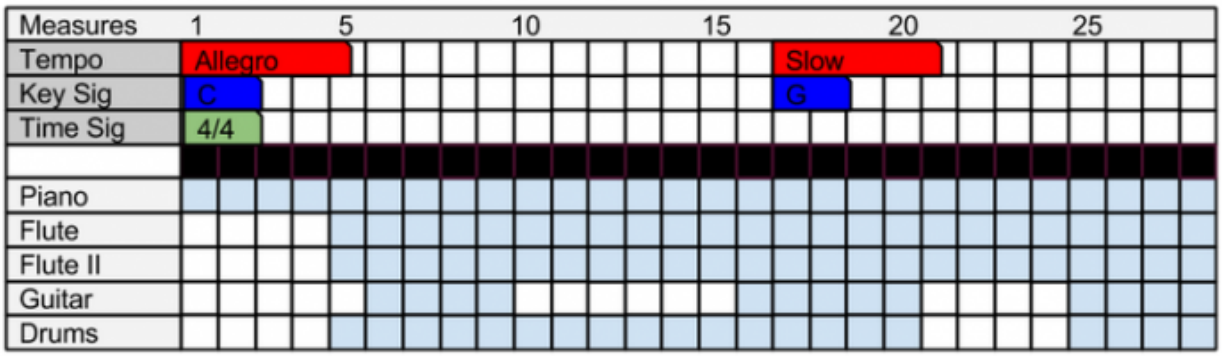
There will be further interaction for rehearsal numbers, tempo changes, key signature changes, time signature changes, and movement tabs. For the first four, the user can *CTRL + LeftClick* on a change or rehearsal number to jump to the next change or rehearsal number. The user can also *CTRL + Shift + LeftClick* to jump to the previous change or rehearsal number. Clicking the movement tabs would display all tempo changes as sub tabs next to the movement number tab. It would also shift the current focus to that movement. The user can then select a tempo change tab to jump to the respective location.

To allow for quick navigation on large scrolls that do not include many of the above features, pressing *CTRL + Left* or *CTRL + Right* will move the timeline focus to the left or right respectively. The amount of measures that the timeline moves will be determined by the number of currently displayed measures. For example, if 20 measures are visible, a shift to the right will move the focus about 15 measures. This will allow for overlap to avoid the user from losing their place. Pressing *CTRL + Shift + Left* or *CTRL + Shift + Right* will move the timeline focus to the beginning or the end of the current movement respectively.

To move around the whole score, there would be a scrollbar at the bottom of the screen. There would also be a scroll bar for the vertical direction to scroll through the instruments added as well. As the user scrolls left and right, the names for the rows would always stay on the screen. As the user scrolls up and down, all the information except the instruments would stay at the top of the screen and only the instruments would change.

Grouping of the instruments would lead to less vertical space taken up. The option to collapse parts would be displayed by a small arrow pointed downwards next to the first item of the group that can be collapsed. Selecting this will change the arrow inwards and collapse the instruments into one row. With existing groups, the name of this row would be given. For user created groups, the row name would be customizable. To create a custom group, the user would hold CTRL and select the individual instruments that they wish to group. Then they would press enter, and it would create a group at the location of the first grouped item.

Another feature would be to display the lyrics of any vocal part. When the user would put their mouse over a square that is a vocalist, a tooltip would be displayed with the lyrics included in that measure. If there are many lyrics, the tooltip would only display a few beats of lyrics to avoid clutter.



Project Schedule

This week-by-week timeline provides a rough guideline of how the project will be done.

12 -- 17 May

Associate myself with the structure of MuseScore, primarily creation and reading of music and creation of tools by using the Developers' Handbook. Practice making small changes to the code and building using Qt Creator. Understand the GitHub workflow used by MuseScore.

17 -- 30 May

Add a new tool for MuseScore named timeline. Using QGridLayout, create a basic timeline, presenting the width as the number of measures and the height as the number of instruments. Implement navigation to the specified measure by clicking the boxes.

31 May -- 6 June

Specifically test the tool with multiple existing pieces on MuseScore and of my own creation. Document my code and plan fixes for any errors in testing that cannot be fixed immediately. If any time is left, begin work on the errors.

7 -- 20 June

Add the time signature, key signature, and tempo to the timeline. Also, add a color change to the boxes where there are notes in the music. Add box highlighting to the current measure and part being edited. Continue careful error checking to avoid large problems.

21 -- 27 June

Test large scores with multiple tempo, key signature, and time signature changes. Handle any graphical issues here. Document all changes.

26 – 30 June

First Evaluation.

Deliverables: Basic timeline with coloring and basic navigation techniques.

28 June -- 11 July

Develop categorization for instruments. Add the collapsing of multiple similar parts into one. Create simple feature for user specific collapsing. Integrate intensity of measures to a darker color for the box. Add tooltips for lyrics in the vocal parts.

12 -- 25 July

Add movement separation, rehearsal numbers, movement tabs, and tempo change subtabs. Add quick movement between tempo, key signature, time signature changes and rehearsal numbers with keyboard shortcuts. Begin final stretch of testing and documentation.

24 – 28 July

Second Evaluations

Deliverables: Timeline with collapsibility, intensity coloring, and quick navigation. General categorization of instruments.

26 July -- 1 August

Test with complex scores with as many features as possible to stress test the feature.

2 -- 16 August

Further refine tests and documentation for the whole project.

Time is left before final submission to allow for unforeseen time issues.

21 – 29 August

Final submission of code

Deliverables: Listed above in deliverables section.

Bio

My name is Joshua Bonn and I am currently a junior at the University of Idaho studying computer science. Just last year, I was also studying Music Theory until time conflicts appeared. My current studies include Qt, C++, artificial intelligence, and analysis of algorithms. My main instrument is cello, and I enjoy

playing piano as well. I have played cello for about 8 years now and piano for about 11. My hobbies include making and eating food, coding, and playing Rocket League. I have created music based applications during my studies. My proudest one is an auto generating four-part chorale based off 18th century rules using genetic algorithms and other local search methods. My current school project is creating an animation builder for light up glasses for the university marching band. This is being built in Qt and C++. Both projects can be found on my GitHub profile. I have also dabbled in development of MuseScore plugins. My main plugin was to automatically detect and display harmony errors as a tool for my music theory class at the university. This was halted as I switched over to the auto generation of music. I have always loved mixing computer science with music, the results are always satisfying and the logic is so unique to music. I can handle steep learning curves by trying many small tests and reading much documentation. I have a solid understanding of editing with MuseScore due to creating individual parts for a handwritten concerto on it as well as auto generating melodies in Python that could be directly loaded into MuseScore. I also have an existing knowledge of programming techniques and programming practices so that I can excel in handling everything that comes with software development. I would love this to be my first step into the open source world as well as a chance to work with those who created MuseScore, which is honestly my favorite score writing program.