GSoC Project Proposal 2019: Palette Accessibility

Profile

Name:	Anand H
Country:	India
School and Degree:	Amritapuri Vishwa Vidyapeetham
Gmail:	anand.h1729@gmail.com
Phone Number:	+91-9400994628
Github:	https://github.com/AnandHemachandran
Telegram Nickname:	anandhemachandran

Synopsis

Musescore is a powerful score-writing software which provides many musicians the ability to create, play and print exquisite scores. The Palette of Musescore is a quintessential tool which provides various elements for editing the scores.

The current palette packs in a lot of elements from all kinds of notations and symbols. Due to the presence of a lot of elements in the palette, it is sometimes challenging to get the required elements at the right time. Therefore a quick UI revision of a palette is needed.

The main changes that I propose to bring to the palette are :

- Making a QTreeWidget to store the Palette Items.
- Implementing a feature to create Custom Shortcut for the Palette Items.
- Creating an Icon View for the Palette.

Benefits to Musescore

Musescore’s Palettes are not accessible to keyboard controls except via the palette search box, also the palette seems to be crumpled with elements which create problems to the users to access the desired elements quickly.

The new Palette is designed using a QTreeWidget and it can provide the Musescore users to quickly access the elements by utilizing the ability of the tree widget to display the palette item in branch and the palette element as the child of each parent branch.

The new palette also provides the users to customize the palette view to different ways

- Icon mode
- List Mode

Deliverables

Redesign the Palette

This will include the creation of two views in which, the list view uses the QTreeView and icon view uses a custom delegate to show palette items in a grid. The position of the Palette Items will also be changed according to the width of the Palette Box.

Optional Extra:

Implementation of custom keyboard shortcuts for the Palette Items

Keyboard shortcuts can be implemented for each of the palette elements as per the user's interest. This feature can be implemented using QMenu to display the option to create a custom keyboard shortcut. The input is taken from the user and then it is compared with the present shortcuts which are already designated for different purposes; if it doesn’t match with the present keyboard shortcut, then a shortcut to the palette is established.

Project Details

The current palette uses QDockWidget and QWidget to display different Palettes on the Palette Box and the Palette.cpp file is used to show the Palette elements in each of the Palette. The ideal implementation would make use of Qt's model-view classes, and QAbstractItemView to create a Category-List View (i.e. a tree view with an icon mode) where the number of columns is able to change dynamically as the user needs. This implementation can be done using the two modules:

- QTreeView.
- QAbstractItemView.

User Interface:

The user interface consists of a Tree widget displaying all the Palette Items as its branch and the palette elements as its leaves. The Palette Items can be viewed in both Grid mode and also List mode.

These views can be implemented by using the following files which are located in the directory `MuseScore/mscore/`

palette.cpp	palette.h
menus.cpp	menus.h
palettebox.cpp	palettebox.h

The shift from icon mode to list mode can be achieved by using the grid-list QPushButton which is situated at the top right corner of the palette box. The width of the palette can be changed by using QGeometry and the icon will rearrange itself along with the change in the grid width.

The palette elements are stored as leaves in each branch (Palette Item) in a QTreeWidget. This can be done by utilizing the menu.cpp file which contains the code for all the palette items.

When the palette item is clicked, the palette elements pop up along with its name, which can also be changed with the help of grid-list icon situated in the top right corner.

The ability to create new workspaces is also present in the new palette design. The workspace can also be edited as per user's need. The information regarding the workspace is provided in a workspace.cpp file.

When the palette items are right-clicked a menu pops-up which can be implemented using a QMenu Class widget. The menu contains options such as creating a Custom Keyboard shortcut, re-arranging the elements and also the ability to change the delete the elements.

Implementation:

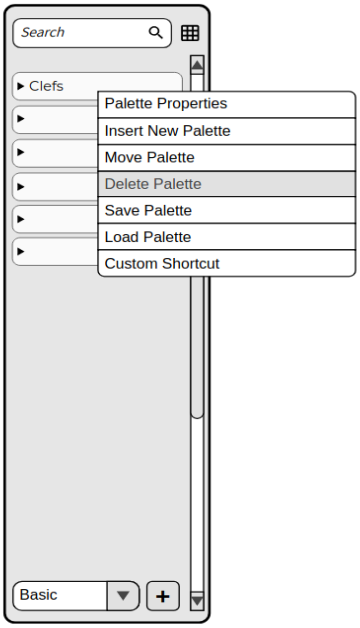
1. List View:

First, a QDockWidget is added. The QDockWidget shall contain the QTreeView to show the PaletteBox as a widget that can be docked in the Main Window.

At the top level, a QTreeView is added. This will give the hierarchical view to the palette as well as manage the scrolling area. Each branch of the TreeView shall contain the Palettes which when clicked pops up to show the different Palette Items in a Grid Layout.

Each Listed Item of the (expanded) QTreeWidgetItem (Palette) will be a QListView or a QListWidget. The icon view of each of the Palette Items can be set using setViewMode(QListView::IconMode) on them. The listView is made using a custom delegate which will handle the icon view and the model of the item which contains the information about the palette item. The information of the palette items for each palette is provided in the file menu.cpp.

The size of the icon in each of the palette can be customized to the user's needs. The Palette Items in each palette can be rearranged using the code setResizeMode(QListView::Adjust), which dynamically arranges the List Item (palette item in Icon View) of the QTreeWidget in a Grid layout with the provided width of the tree view.

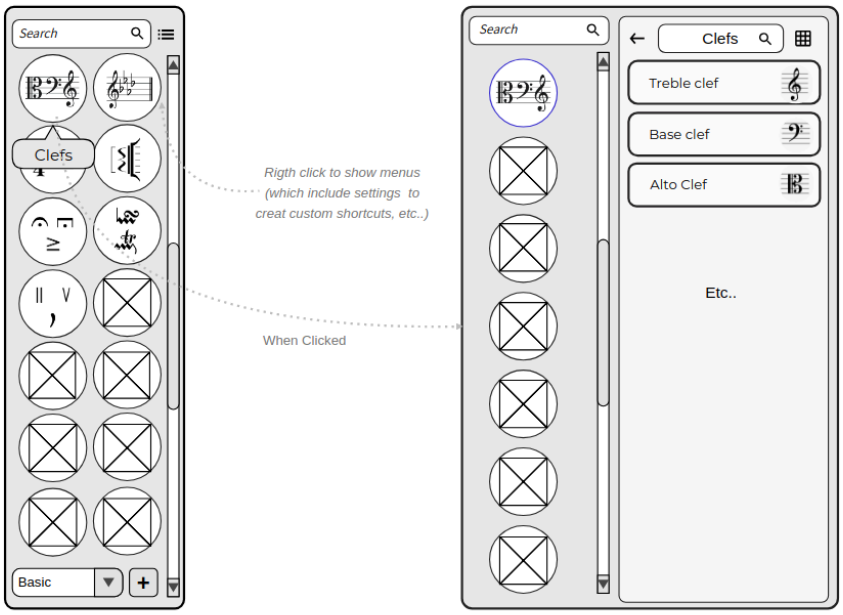


Custom Keyboard Shortcut Implementation

Each of the List Items (Palette) contains certain options. The menu can be opened by right-clicking the Palette. The options are created using the QMenu. The new option included is called the “Custom Shortcut” which helps you to assign a keyboard Shortcut to the Palette Item. This is then cross-checked with the shortcuts already assigned for other purposes. This is done by utilizing the shortcut.cpp file.

2. Icon View:

This view can be achieved using Qt’s Custom Delegate on QListWidget. The ListWidget is shown as an icon as provided in the mockup using the code setViewMode(QListView::IconMode).



Project Schedules

Week No	Timeframe		Plans
	From	To	
	May 6	May 26	Community Bonding.
1.	May 27	June 2	Start working with the UI and make the basic QTreeWidget.
2.	June 3	June 9	Implement each palette element one by one by creating a QListWidgetItem.
3.	June 10	June 23	Work on the layout of the QTreeWidget and implement the signal and slots for the palette by making use of the information provided in the menu.cpp for each palette.
4.	June 24	June 30	Phase 1 Evaluation.
5.	July 1	July 7	Using Custom Delegates to add Icons to the Palette Items
6.	July 8	July 14	Add the ability to change the view, size of the Palette Box and also the ability to rearrange and dynamically allocate the palette icons.
7.	July 15	July 21	Add the menu option for each palette Item using the QMenu Widget. Add the feature to create a Custom Shortcut for the Palette Item.
8.	July 22	July 28	Phase 2 Evaluation.
9.	July 29	August 4	Continue working on the Keyboard Shortcut Implementation.
10.	August 5	August 11	Make a final revision in the new UI.
11.	August 12	August 19	Complete the documentation. Work on the bugs, if any are present.
12.	August 19	August 26	Final Evaluation.

Bio

I am Anand H, an undergraduate Computer Science student at Amrita Vishwa Vidyapeetham. My keen interest in music and Open-Source Software are the two major reason why I am contributing to this software. I have played keyboard in my free time and also have attended Trinity College of music till fifth grade for my keyboard lessons. The time I have spent to learn music has provided me with substantial knowledge on notes and notations of a music sheet. I have used Musescore to create and practice new scores, this provided me with the foundation to work for this organization. I always enjoyed using this software since its powerful, elegant and most importantly it is an Open Source Software. I am really excited to contribute to this organization.

The reason why I selected Palette Accessibility as my project is that I have always found the palette elements to be a bit challenging to acquire and an interesting segment to develop on since it is vital for any artist to get the required notations in the right time. I hope, my knowledge of Qt and C++ would be resourceful for the organization. For the past 4 weeks, I have dedicated myself to get familiar with the codebase and went through different documentation on Qt to get to understand as many concepts as possible on QTreeWidgets, Delegate Class and Model/View Programming and I am pretty sure I will be able to work on this project.

My work time would be around 40-45 hrs per week except for the time period during the community bonding which is when my Semester exams will be conducted. Even if I won't get GSoC I will still continue to contribute to this software since this is my ideal score-writing software.

Work Done till now:

Submitted PR:

- Update Play Panel
- fix #270765:Update Layout for Plugin-Manager
- fix #281810:Fix Label Truncation
- Update Timeline Tempo

Qt Works:

- https://github.com/AnandHemachandran/Qt-Projects
- https://github.com/AnandHemachandran/QT-Calculator