DJNorm .MP3 Music Library

Darrell, Fiacre, Tazia

ITCS 3112

December 10th, 2017

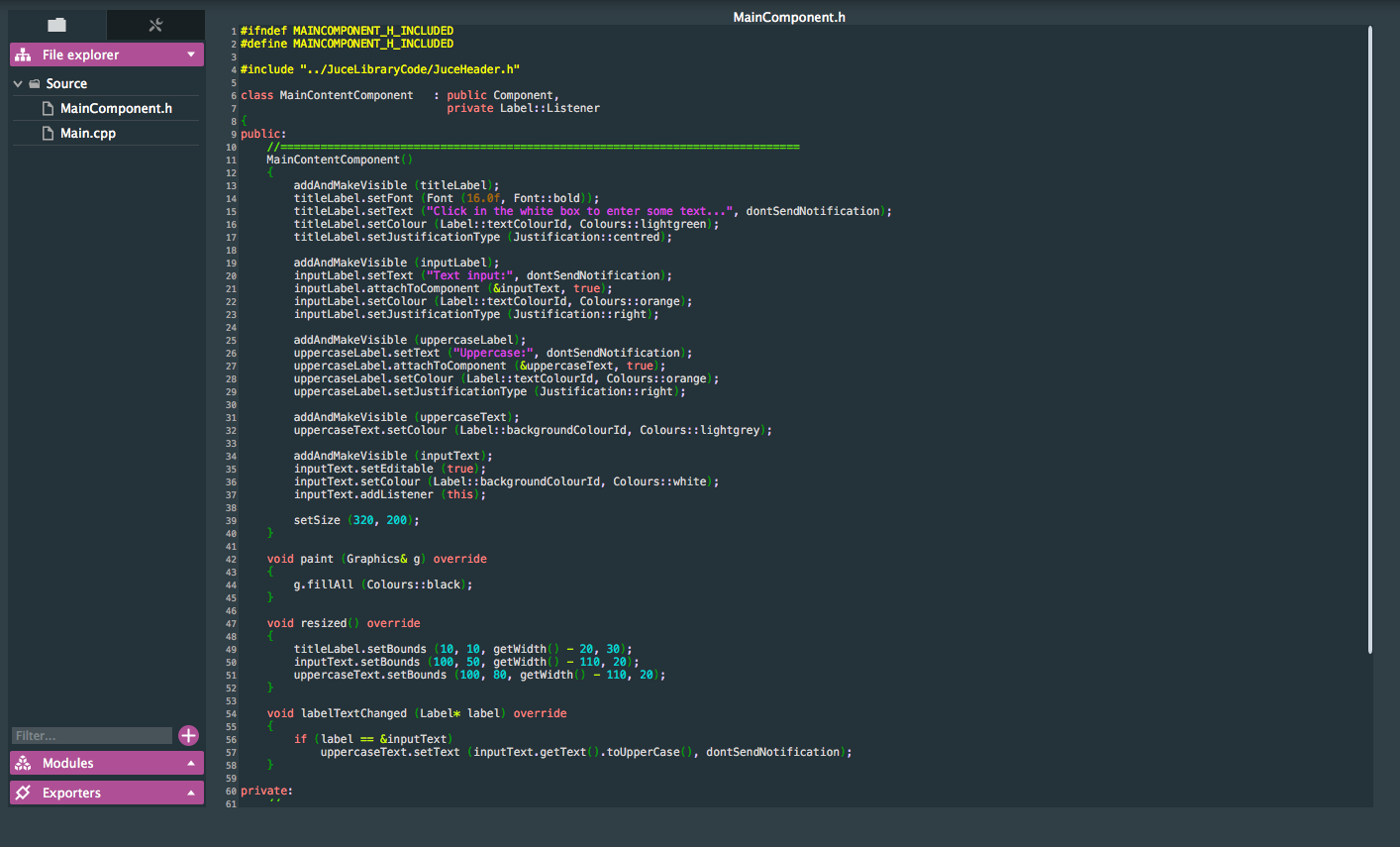
## 

## Outline

1. Introduction
2. Design
3. User Interface/Features
4. Testing
5. Issues/Status
6. Conclusion

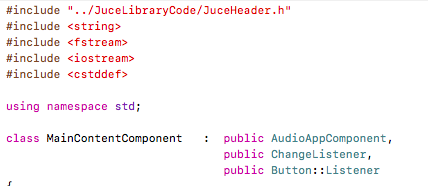
## Introduction:

DJNorm is a music library application that allows the user some functionalities such as: viewing the library, sorting by the artists alphabetically, sorting by the songs alphabetically, sorting by genre alphabetically, and searching the dataset.

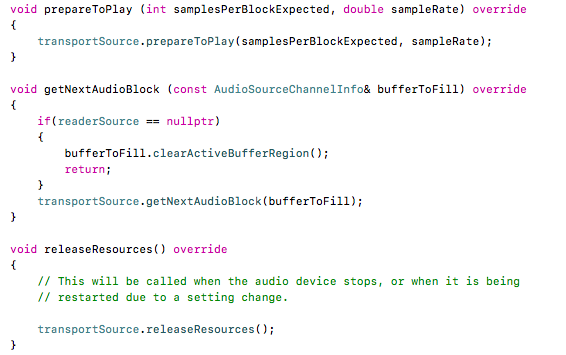


## Design

The JUCE library contains various classes that can be used to create visual effects for audio based applications. Our MainContentComponent class that is called by our main() derives the AudioAppComponent, ChangeListener, and the Button::Lister classes inside of the JUCE library.



The AudioAppComponent inherits from the AudioSource class which contains the object that allows streaming between the audioappcomponent and the audio hardware device. The task forming multiple audio objects together to play a song is simplified by utilizing the getNextAudioBlock() function in the class. After creating the GUI inside of the MainContentComponent class we then added listeners to each of the buttons. We included a deconstructor to shut down the audio when closed. The remaining functions such as the getNextAudioBlock( ) function are included in the C++ library and handle the audio for us.



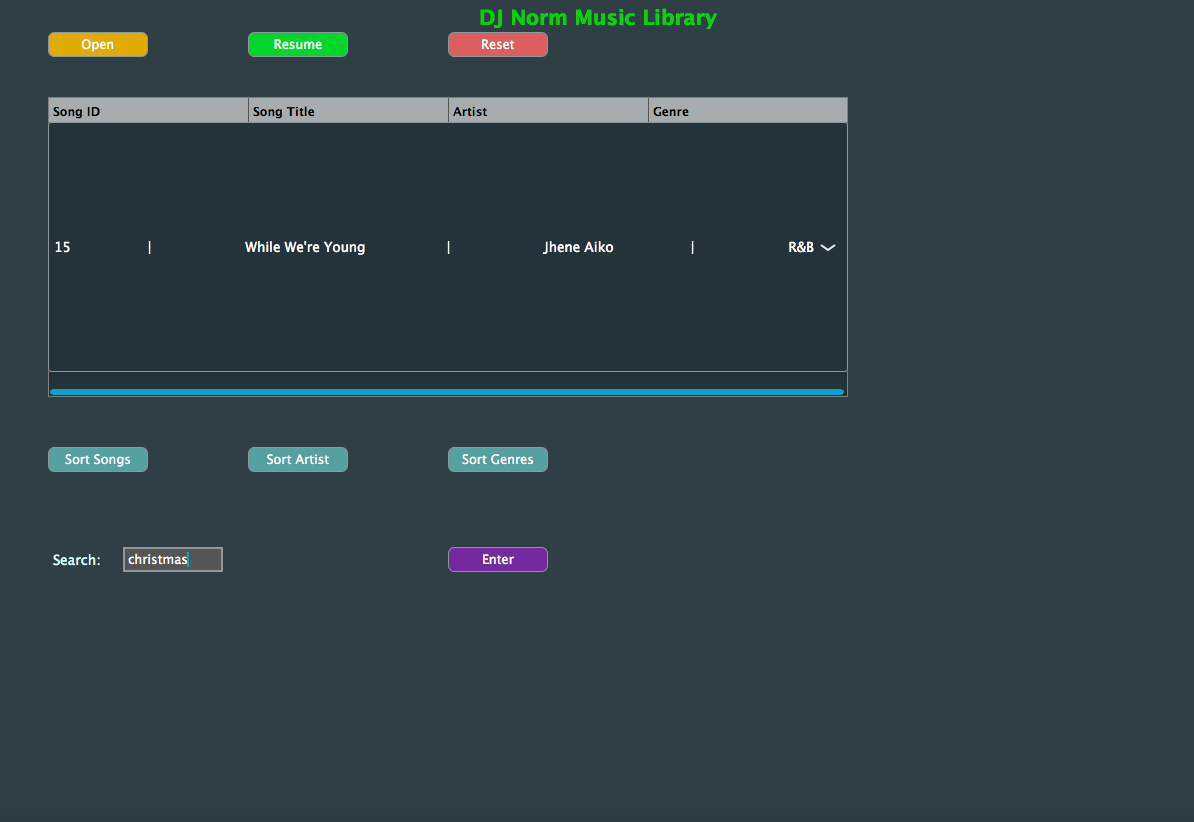
We then created states that change are selected in a switch function to enable the user to effectively interact with the application when opening files, pausing, or playing audio.



We also included various string manipulation methods ranging from loading the data into the table, manipulating the data string to better the visualization in the table, handling button click events, and for searching the data inside the table.

## User Interface/Features

Utilizing the C++ library, JUCE, we were successfully able to create a GUI using its components. First we created a resizable window with a Main Content Component that holds child components within itself. This window can be resized, closed, and minimized on the desktop. We then created a label component that displays the title of our application at the top-center location of the window. We set the justification to centered so that it is always displayed in the center of the window when it is adjusted. We then added open, play, stop, and sort button components that change the state of the program according to the users interactions with them. When a user opens a musical file within the program directory by clicking the “open” button the others are then enabled allowing the user to interact with them. The user can then play, pause, and reset the selected audio input. In between the button components a table with headers and combo box has been added. This table contains column headers that identify the song id, song title, artist, and song’s genre of songs. A user can click the combo box and select an item from the list. The combo box then displays the user’s selection in the table. The application also includes a search function that accepts user input through a text box. After entering a string into the box the user then clicks the search function and a search through the database is performed. If the search finds a match it selects the combobox item where the string is found and sets it to the current selection in the table.

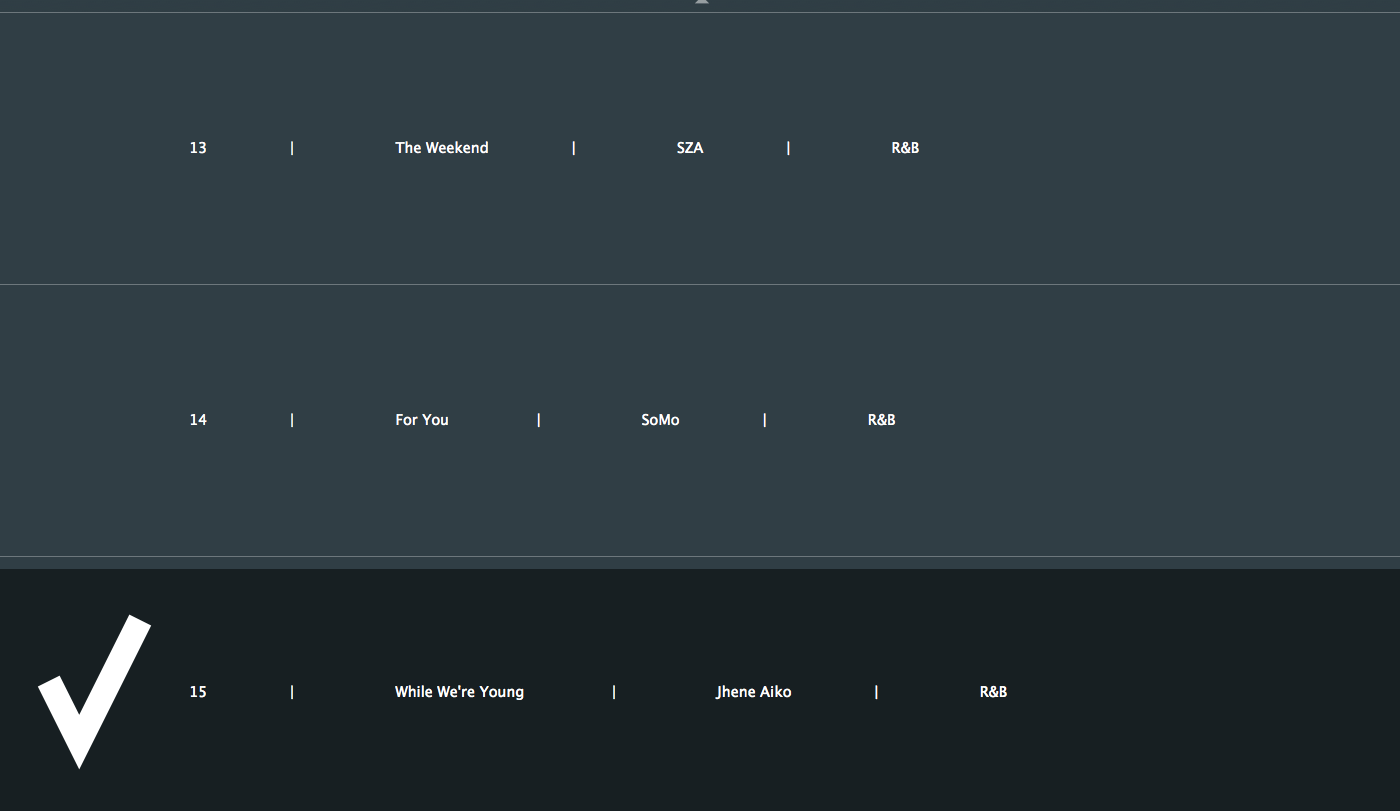


## Testing

For testing purposes, we used a list of a variety songs and files containing the artists and genre of those songs. We also made sure that the JUCE library functionality works cross-platform.

## Issues/Status

Some of the issues we encountered included being able to read the library without exclusively including it in an excel document. We also faced the challenge of being able to access song file names in order to match with the song ID’s inside of the dataset. Because of this we also could not allow the music to loop through the file directory to continuously play music. There was also limited resources available in the JUCE library for working with audio in C++. Therefore, we focused on making sure the applications primary functions worked: the user is able to interact with an interface to play audio files and search through a sorted data set.



## Conclusion

Developing the application was challenging but due to use of the C++ library JUCE, it made the process easier. We have the basic functionalities working and successfully created a visually appealing user interface to integrate with it. The potential in using the JUCE library for audio applications is huge and is a great environment to start with a basic understanding of object oriented programming in C++.