# 

|  |  |
| --- | --- |
| **Student Name:** | HAN YAN XIN |
| **Date Submitted:** | 15th March 2021 |
| **Degree Title:** | BSc in Computer Science |
| **Local Institution:** | Singapore Institute of Management |
| **Student ID:** | 190428158 |

Contents

[R1: 3](#_Toc66141881)

[R1A: 3](#_Toc66141882)

[R1B&R1C&R1D: 5](#_Toc66141883)

[R2: 6](#_Toc66141884)

[R2A: 6](#_Toc66141885)

[R2B: 7](#_Toc66141886)

[R3: 8](#_Toc66141887)

[Introduction: list’s button set in playlist Component: 8](#_Toc66141888)

[R3A&R3B: 9](#_Toc66141889)

[R3C: 12](#_Toc66141890)

[R3D: 12](#_Toc66141891)

[R3E: 14](#_Toc66141892)

[R4: 17](#_Toc66141893)

[R4A&R4B&R4C: 17](#_Toc66141894)

# R1:

## R1A:



Figure 1.1 DeckGUI header

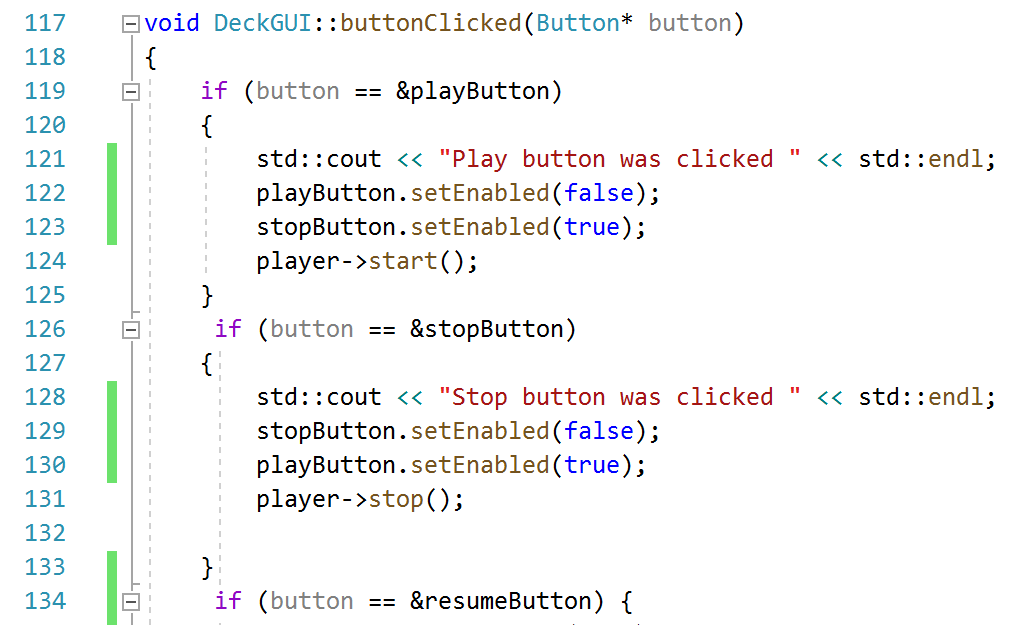


Figure 1.2 ButtomClick function to play



Figure 1.3 The Deck GUI layout

Through the playlist component from R3, the play & stop controller can load (load function have been moved to the playlist component currently) & play the files and draw the waveform of the track by clicking the button of “to Deck1” and “to Deck2” in the same time. In the figure 1.2, the player can start and stop (line 122, 127) the track in the function of ButtonClicked, the player pointed to the start() or stop() function and play the music. When the user clicks the button, the function can check which Text button have been clicked and call the relative function. In the line 122,123The user cannot click the play button when the music is played (stop button is the same in line 129, 130) so that the application can remind the user that the music have been started.

## R1B&R1C&R1D:

In the figure 1.2, two rotate sliders with the labels have been shown in the layout from DeckGUI class. Those two sliders can set the volume and speed of the track which is loaded.



Figure 1.4 DeckGUI constructor

In the constructor of DeckGUI class, the volume and speed sliders and their label have been declared as the variables. Those two components have been set as rotate slider. By two DeckGUIs in the window, the two tracks can play music in the same time with changing the volume and speed of two tracks.

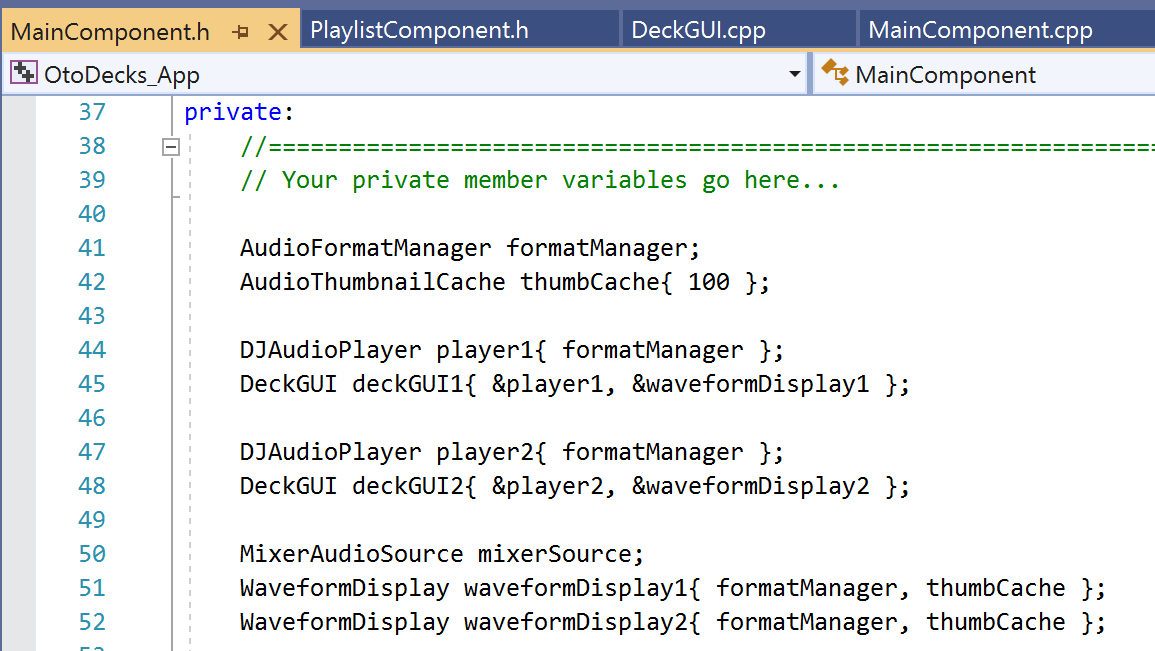


Figure 1.5. The main component header

In the figure 1.4, there are two DeckGUIs which contain the DJAudioPlayers and waveform display class variables. Through the two DeckGUI classes, two tracks can be played in the same time for fulfill the R1B.

# R2:

## R2A:



Figure 2.1 getLookAndFeel function in the playlist Component constructor

By using the getLookAndFeel function in the constructor, the table component can be customized. The searching box’s text and background’s colour have been set in the RGB mode in the line 41,42 (the same as the table component background in line 44).

The getLookAndFeel function also works in the DeckGUI component in the figure 1.4. line 38-42, all sliders and Textbuttons in the application have been set the specific color. The set also can work in the list box component in the playlistCoponent class, the buttons in the list have been set to the same colors as the buttons in the DeckGUI class.

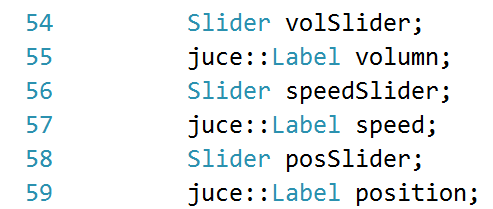


Figure 2.2. The sliders and labels

In the figure 1.3, the three sliders are with the attached labels. The labels are defined in the DeckGUI header file with the slider(the figure 2.2) and the labels are set to attach the corresponding slider in figure 1.4, line 51-53 (the variable of Boolean can decided the labels’ position with the slider). Though the volume and speed sliders are rotate slider, the label and textboxes are set to the different location which is different with the position slider (figure 1.4, line 27-30).

## R2B:

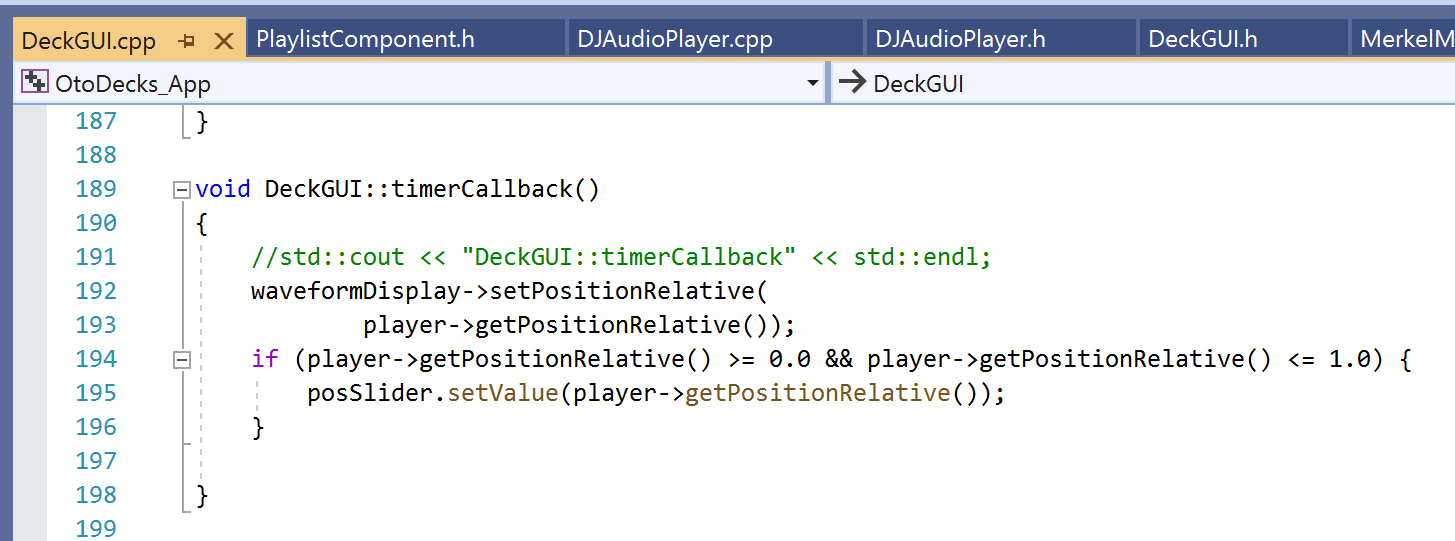


Figure 2.3 The timeCallback function in DeckGUI

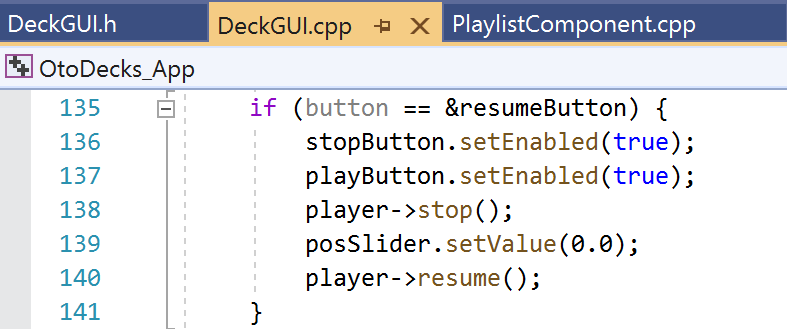
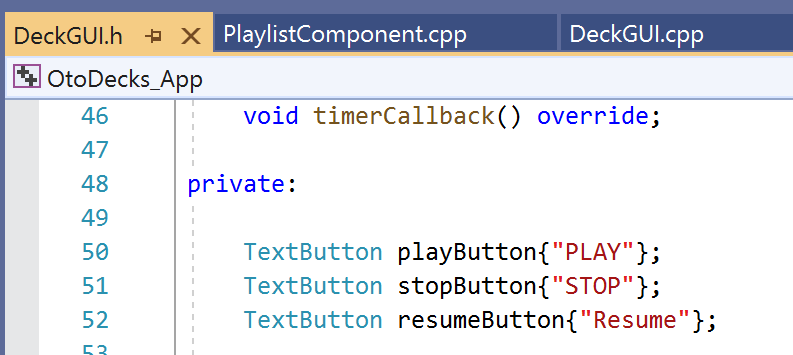


Figure 2.4 The Resume buttons in DeckGUI header and cpp

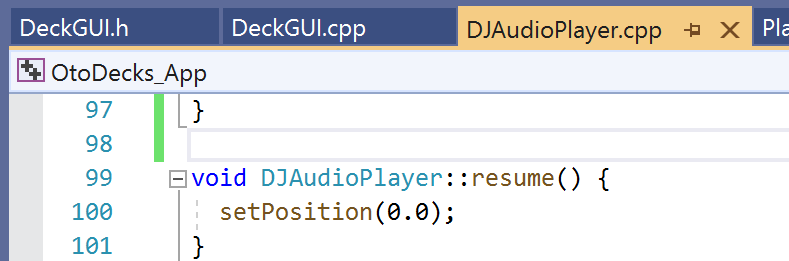


Figure 2.5 The resume function of DJAudioPlayer

Through the timeCallBack, the position slider’s thumb can move the corresponding position. (the if in line 194 can avoid the error of null value to the posSlider). The position slider can play the role of playback bar in the application, which can be seen in the figure 2.6. Users can drag the thumb or click the position on the slider’s track to set the play progress.

The Resume button under the stop button can set the track’s position back to 0 and stop the music. The code which is under the ButtonClicked can be seen in Figure 2.4 right side.



Figure 2.6 The position slider, the waveform and the Resume button

# R3:

### Introduction: list’s button set in playlist Component:



Figure 3.0.1 The component ID set of each button

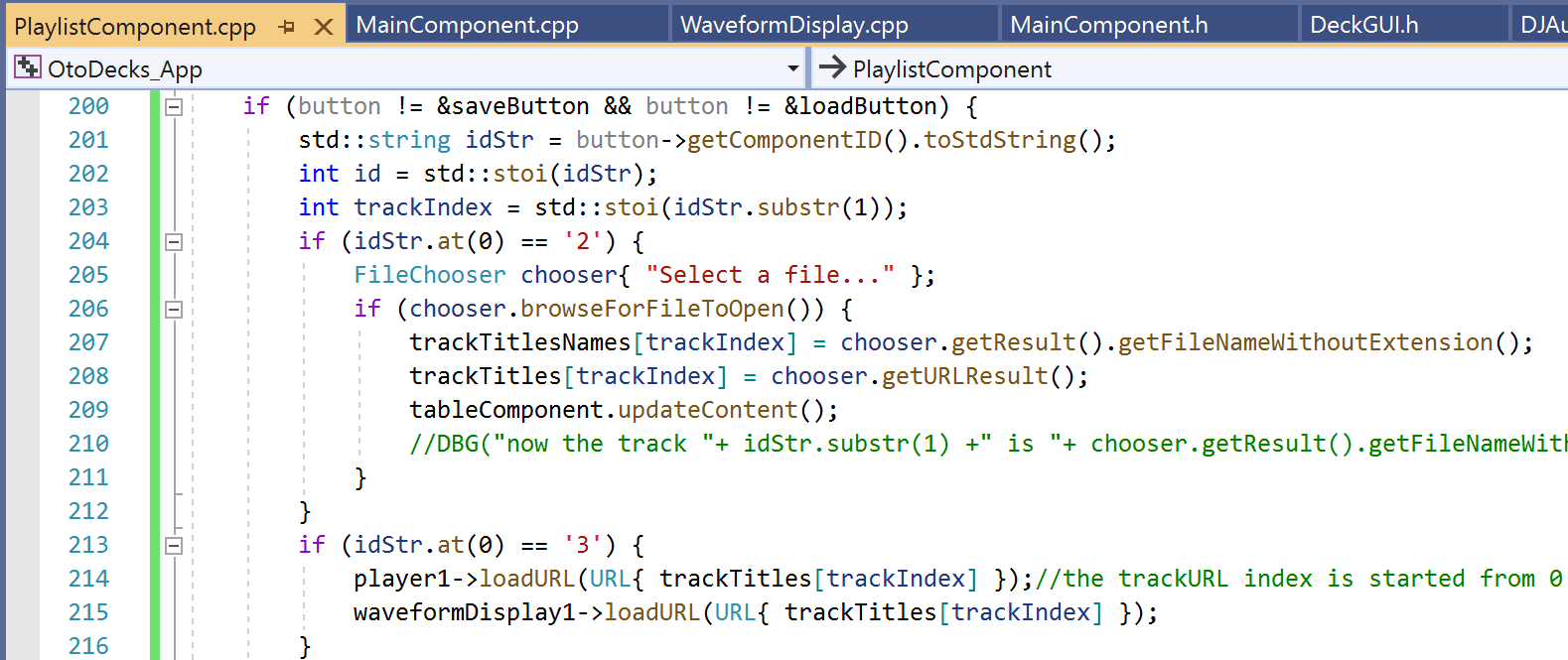


Figure 3.0.2 The button id reading in ButtonClicked function

The button set in the list box is different with the other text button. The address pointer for buttons cannot work in the tableListBox. The application has set the id for each button with the corresponding Row Number and Column Number (if the button is on row 4 and at column 2, the button is the load button for the track which is index 4 from trackURL vector and trackName Vector). In Figure 3.0.2 line 201, the idStr is the string of button’s id. The trackIndex is the string id’s substring from second index (for example, id = “412”, the index is 12, “4” is for telling the button’s function. Application can obtain the integer by the function of std::stoi(std::string ), at line 202,203 ).

## R3A&R3B:

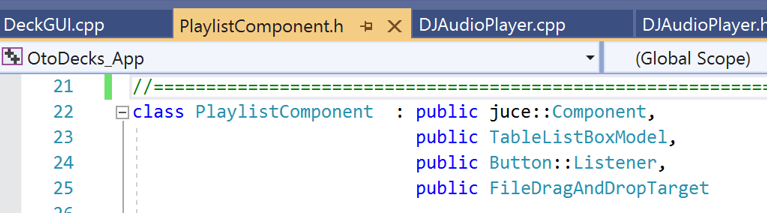


Figure 3.1 The header file of playlistComponent

The process of adding the track have been separated into two steps: The pop-up and file drop

The playlistComponent inherited the FileDragAndDropTarget, so that the user can drag the file into the playlist part in the window to add the track. The user does not need to load the file with the file chooser one by one, the adding of dragging can be more effective.

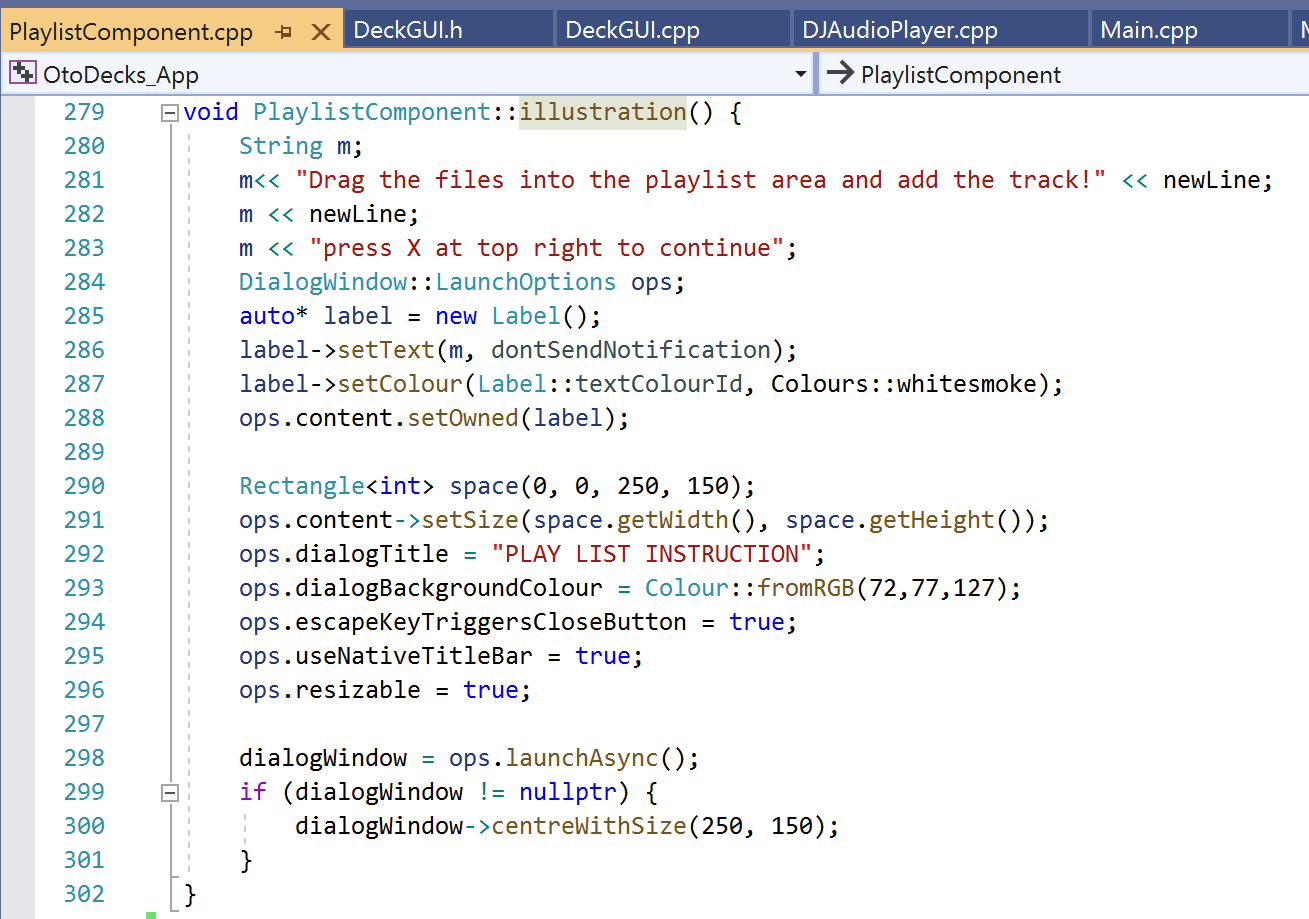


Figure 3.2 The illustration function in playlistComponent

For the instruction of teaching user how to add the track, the illustration function can show pop-up after the user running the application. The user needs to close the window so that they can continue. The pop-up background’s color has been set in line 293. The string has been set in the center of the window and paragraph started from left.



Figure 3.3. The pop-up

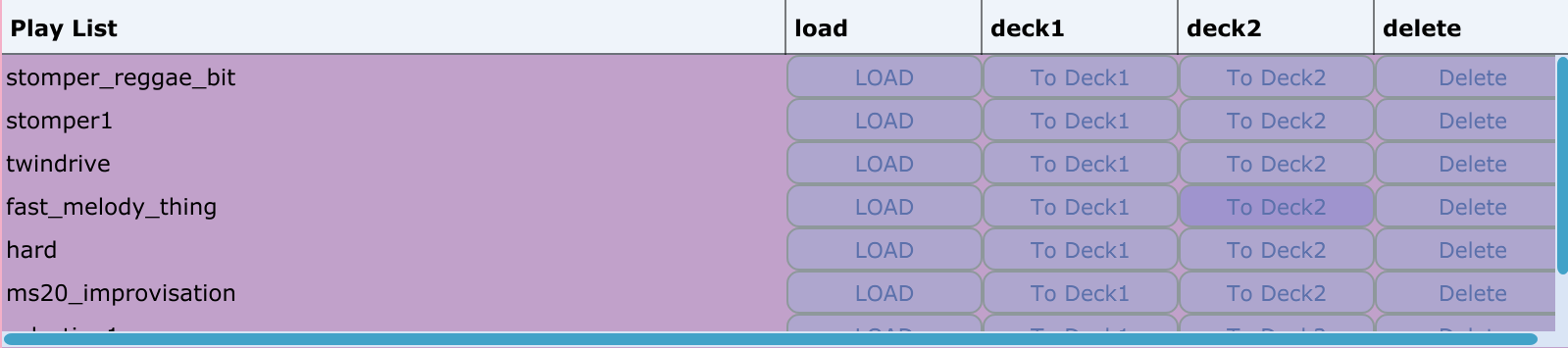


Figure 3.4 The library at the bottom of the window

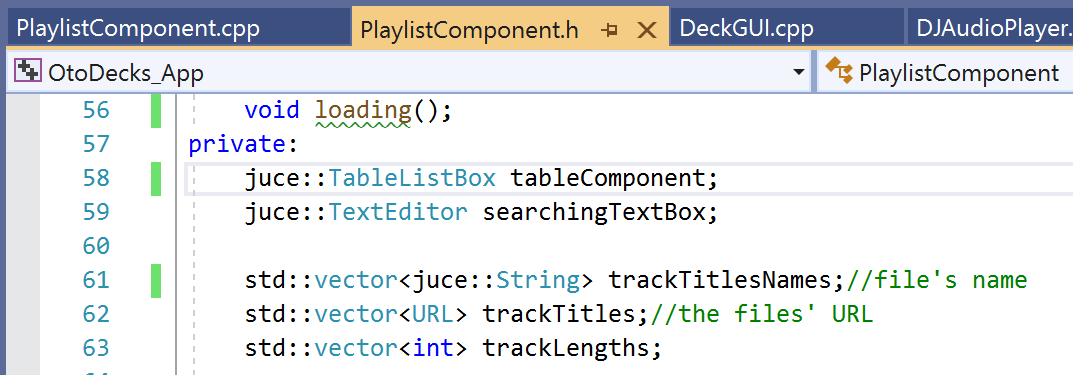


Figure 3.5 The vectors of trackTitleNames and trackTitle

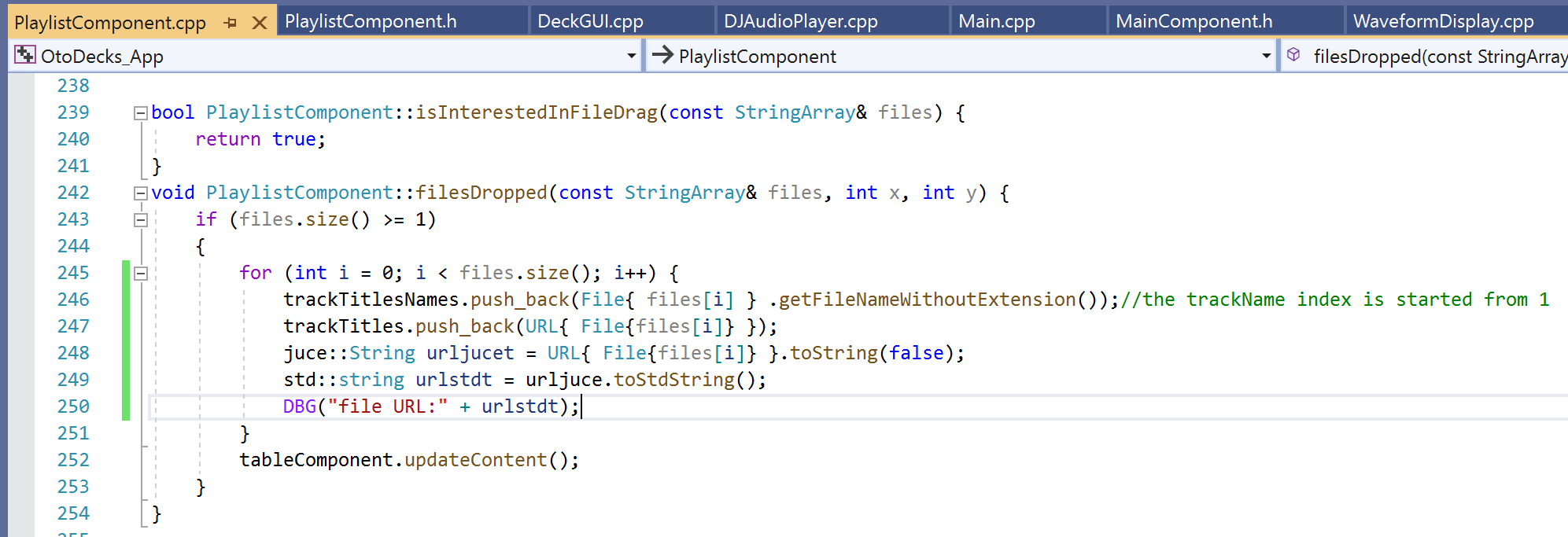


Figure 3.6 The function of adding tracks

For R3B, after the file adding, the playlist can show the files’ names and four buttons. The load allows users to change current row’s data which included the file name and relative URL.

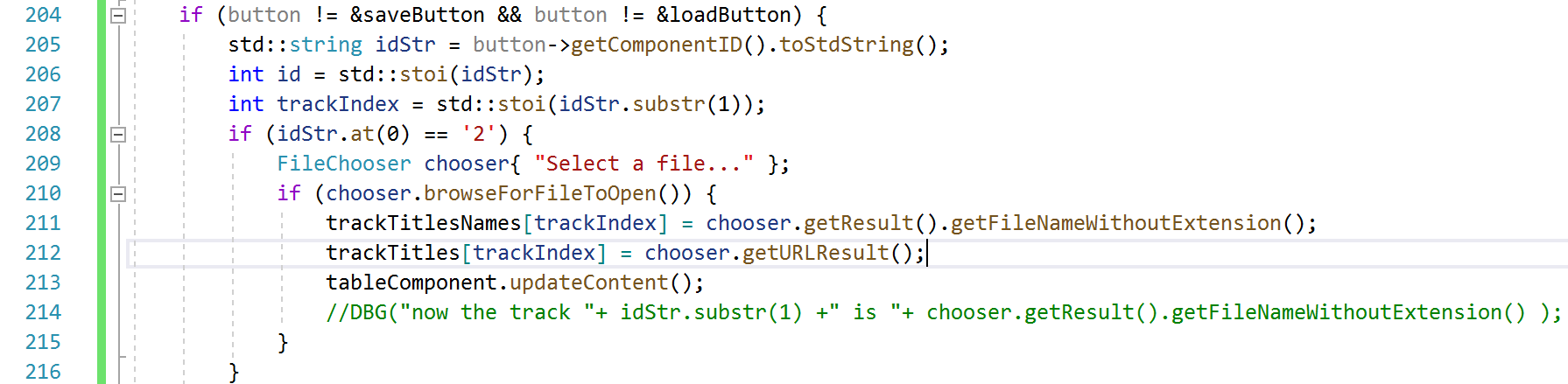


Figure 3.7 The load button

The playlist can update the content after user’s adding and deleting. The code of deleting the row and data can be seen in figure 3.5, through the erase function of the vector, the index and row can move forward automatically. The conflict of indexes cannot occur. (For example, after the erase, the data of index 3 in trackTitle and trackTiTleName on the row 3 is removed, the origin row 4 data which is on index 4 can be moved to row 3, and the strings in two vectors also can move forward to index 3. The index 3’s position will not be null. The erase function lets the removing become simpler.)

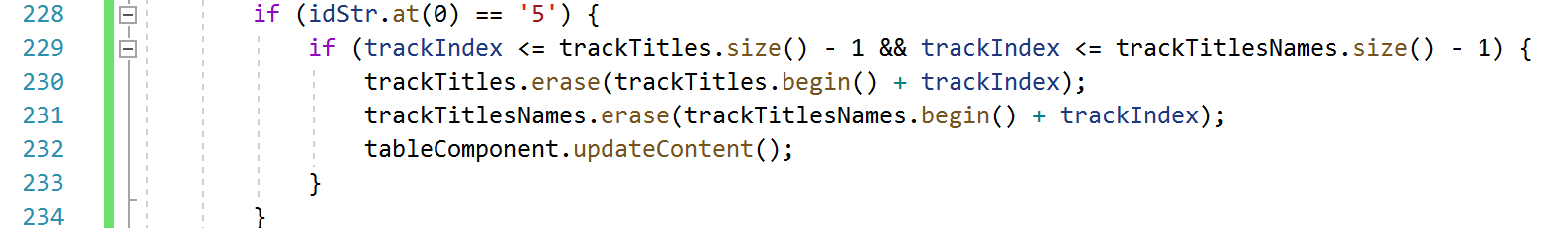


Figure 3.8 The delete under ButtonClicked function in playlistComponent cpp

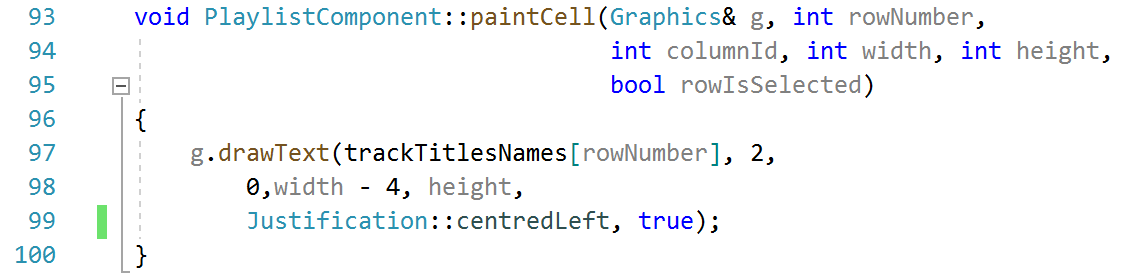
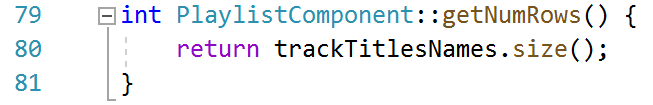


Figure 3.9 The two list box functions related to the trackTitleName

## R3C:

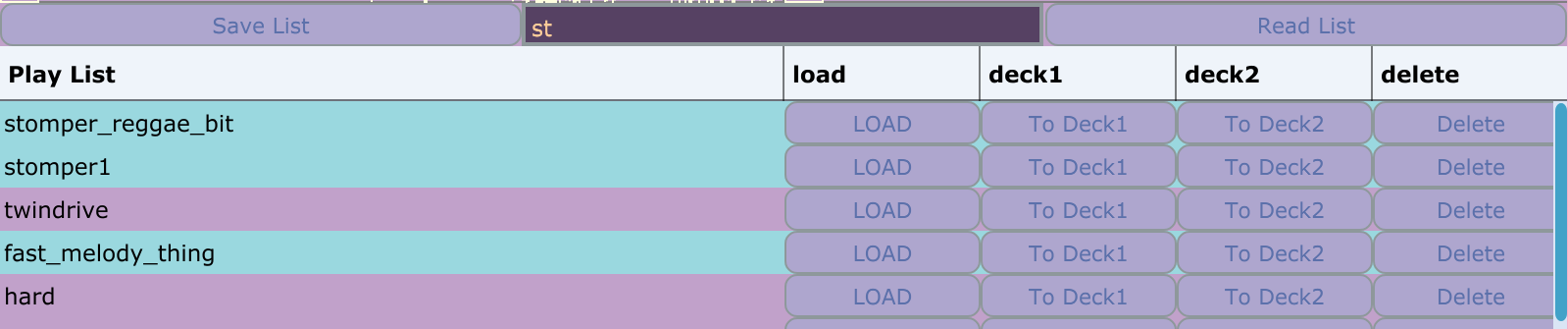


Figure 3.10 The selected rows of the search result

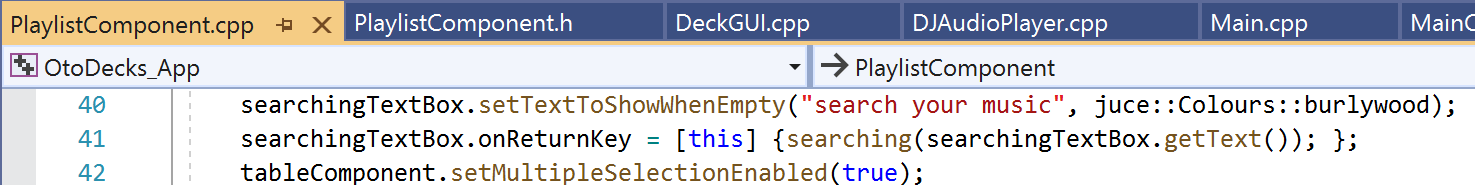


Figure 3.11 The search box’s text and functions’ set

In the figure 3.10, the rows can be selected as the search result. The playlist selects the data which contains the keyword entered by the user. The onReturnKey function of textbox can call the function of searching with the getText() which is shown in the figure 3.12 and execute the code of selecting rows (line 41-42).

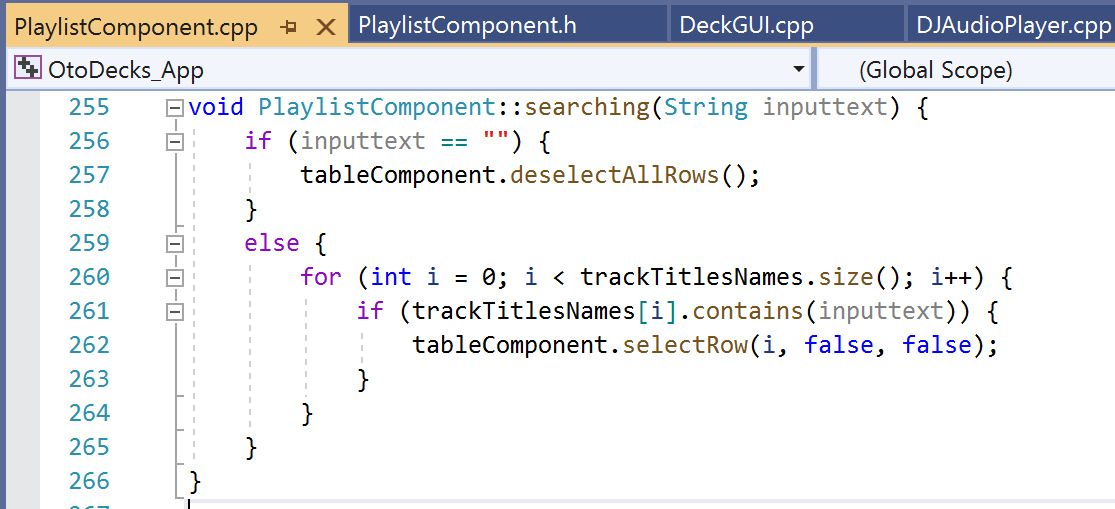
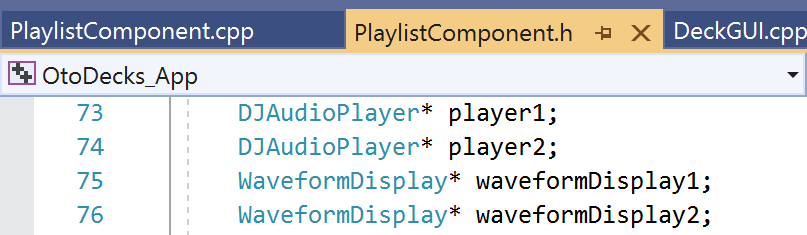


Figure 3.12 The searching function related to the searchingTextBox

In the searching function of figure 3.12, if user enter null, the table deselect all rows. In the line 260 with the else, the for loop can loop through the files’ name and select the rows which contain the input text by users (line 261, juce::string.contain(std::string)). In the figure 3.11 line 42, the table is allowed to select multiple rows, so that the searching function can highlight the result rows.

## R3D:



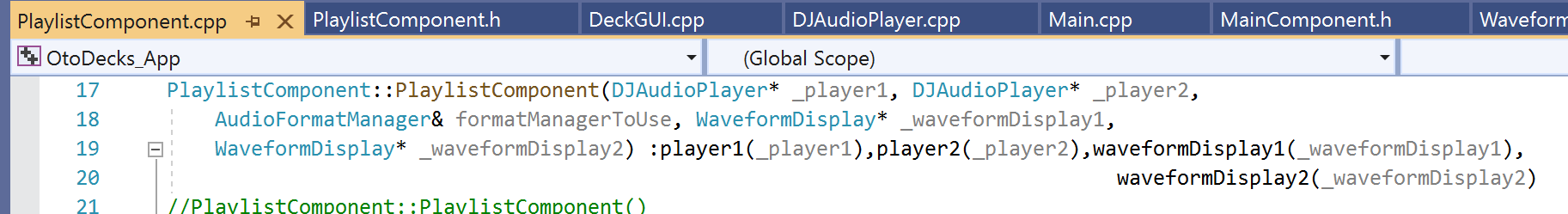


Figure 3.14 The playlistComponent’s pointer and constructor

For the DeckGUI and waveform loading, the DJAudioPlayer and WaveformDisplays’ pointer have been declared in the playlistComponent header as private variables. Those four pointer point to the addresses of two players and two waveforms from the main component. Thanks to the pointer of c++, it is not necessarily to create new variables of those two classes and directly point to classes which existed already. The declaration of those four pointers are similar with the pointer which is in the DeckGUI to DJAudioPlayer from the starter code (figure 3.15).

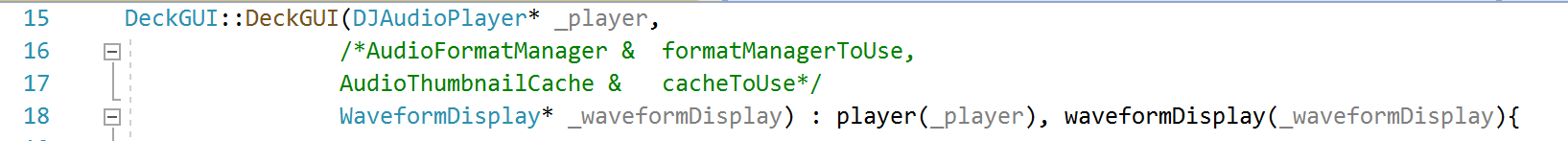


Figure 3.15 The pointer in the argument of DeckGUI

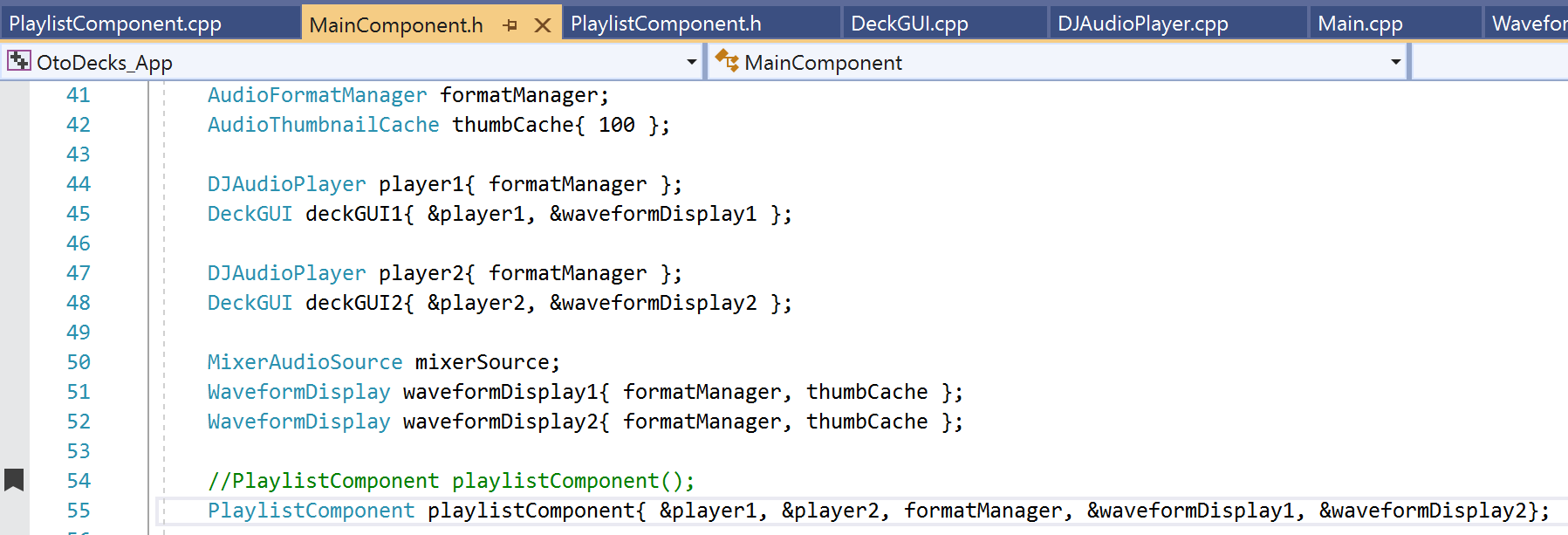


Figure 3.16 the variables of Main Component

The waveformDisplay have been moved out from DeckGUI and moved into the main component for further playlist loading.

The &player1, &player2, &waveformDisplay1 and &waveformDisplay2 in the playlistComponent arguments is the address of the variables in line 44,47, 51 and 52.

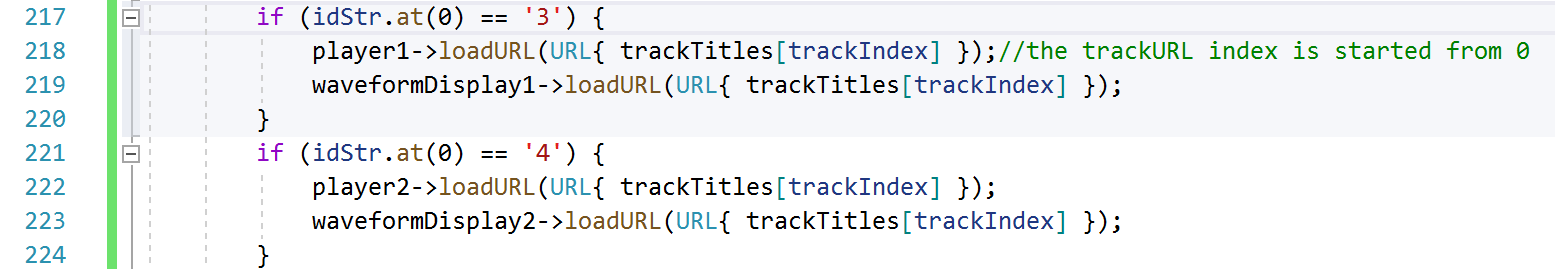


Figure 3.17 The function of loading to two DeckGUI

With the button of load to Deck1/Deck2 at line 217 and 221, the player can point to the loadURL function from DJAudioPlayer and activate the DeckGUI from playlist directly (waveform is the same).

## R3E:

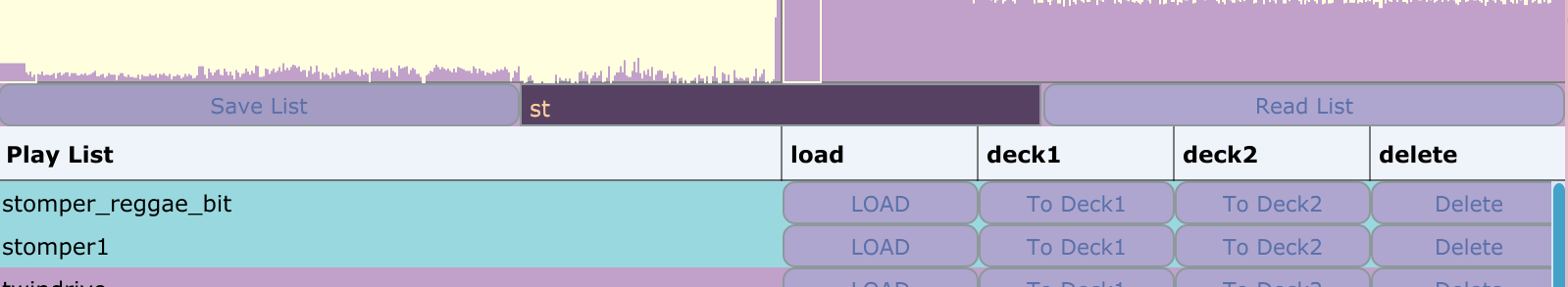


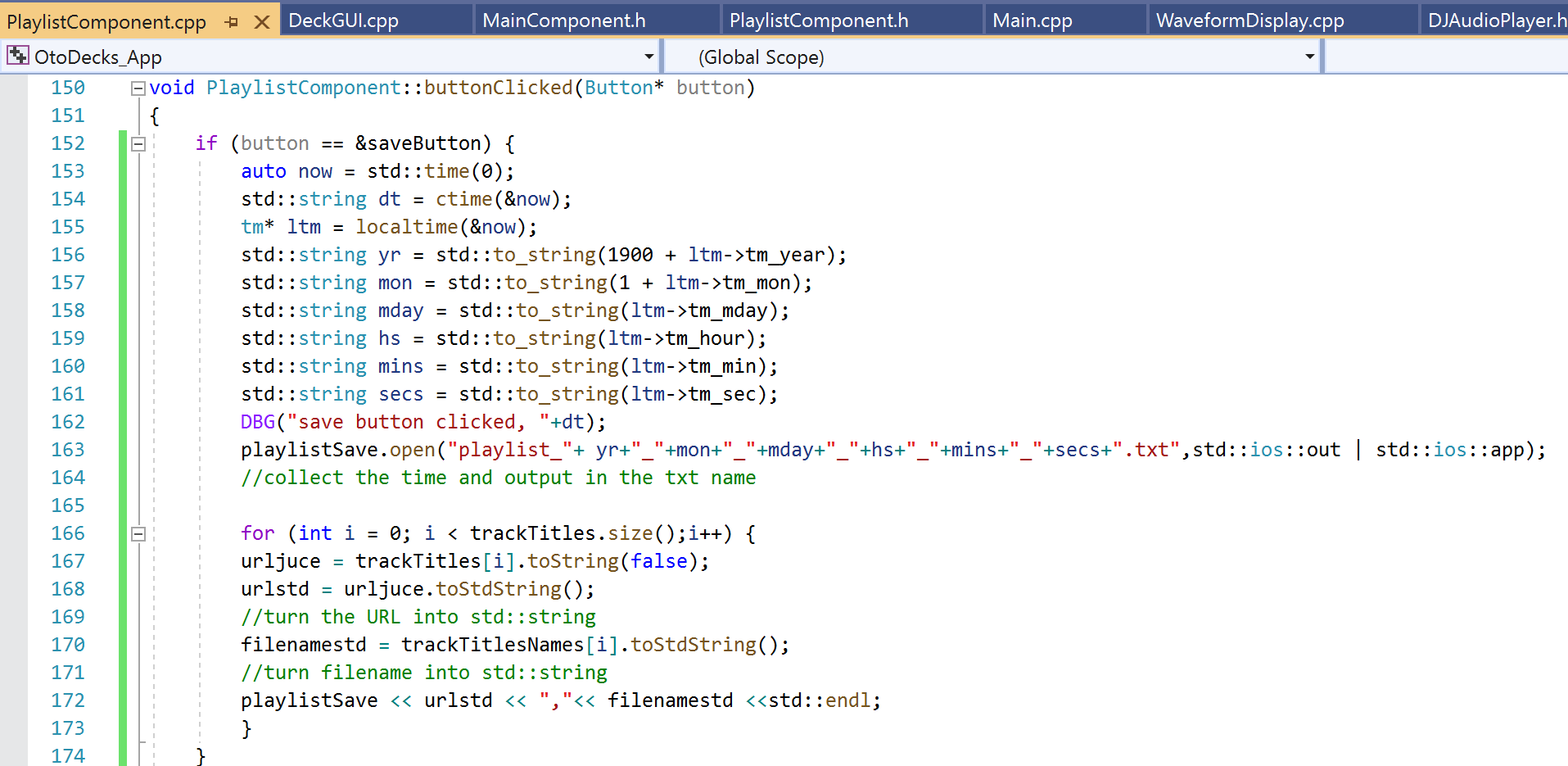
Figure 3.18 The save list and read list text button in the playlist area.

Figure 3.19 output the list

The saveButton as a new text button has been declared in the playlistComponent header and added the listener. In the figure 3.19 line 153~163 is the process of output the file name. The application can collect the time in the system return the value of time from 1970 to now 2021. By using the std::to\_String function, the t\_time variables can be transferred into multiples std::string for placing the file name in the lines 156~161. The files’ name saved from the application can be seen in figure 3.20.

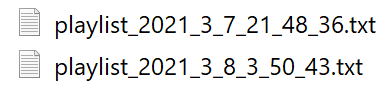


Figure 3.20 The playlist’s name

The trackTitle is the vector of storing URL, for outputting the string, the url need to URL -> JUCE::String -> std::string (line 167,168 in figure 3.19). File name is similar in line 170 (JUCE::String -> std::string). The ofstream can loop through the two vector and print them line by line.

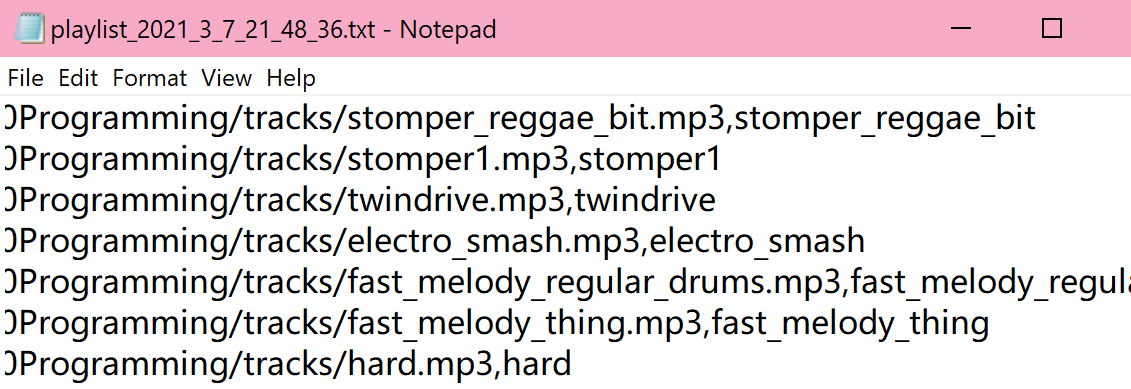


Figure 3.21 The txt’s line printed by application

The user can load the txt file by the loadButton in figure 3.18 “Read List”. Figure 3.22 has shown the process of loading the txt list file.



Figure 3.22 load button clicked



Figure 3.23 tokenise function in playlistComponent

In the loading function of figure 3.22, the application clears two vectors and read the file with ifstream in lines. The std::getLine can read the txt file one line at a time. One line consists of two strings. The first string is the URL, and the second string is the file name.

*file:///C%3A/Users/Documents/CM2005%20ObjectOriented%20Programming/tracks/ms20\_improvisation.mp3,ms20\_improvisation*

Example of the line from txt file.

The std::string is separated by the “,”, with the function of tokenise which is similar to the mid-term file reading. The tokenise function returns the std::string vector (line 301 in Figure 3.23), the token[0] is the URL in std::string, token[1] is the trackTitleName for the row painting in std::string. Thanks to the JUCE::String is compatible with the std::string. The push\_back of trackTitleName can succeed with the std::string (line 190 in figure 3.22). The URL push\_back can be executed by URL { std::string } (line 189 in Figure 3.23). The playlist can be loaded and work successfully with the correct URL and String variables.

# R4:

## R4A&R4B&R4C:



Figure 4.1 The layout of whole application

The layout of DeckGUI has been introduced in the R1 (the rotate slider, its thumbs, tracks of sliders and the color set) The background of the waveformDisplay has been set in the DeckGUI by getLookandFeel() function. The waveform color is set as lightyellow from Colours in line 40 of figure 4.2, and the rectangle’s colour of showing the position is set as aqua in the waveform area.

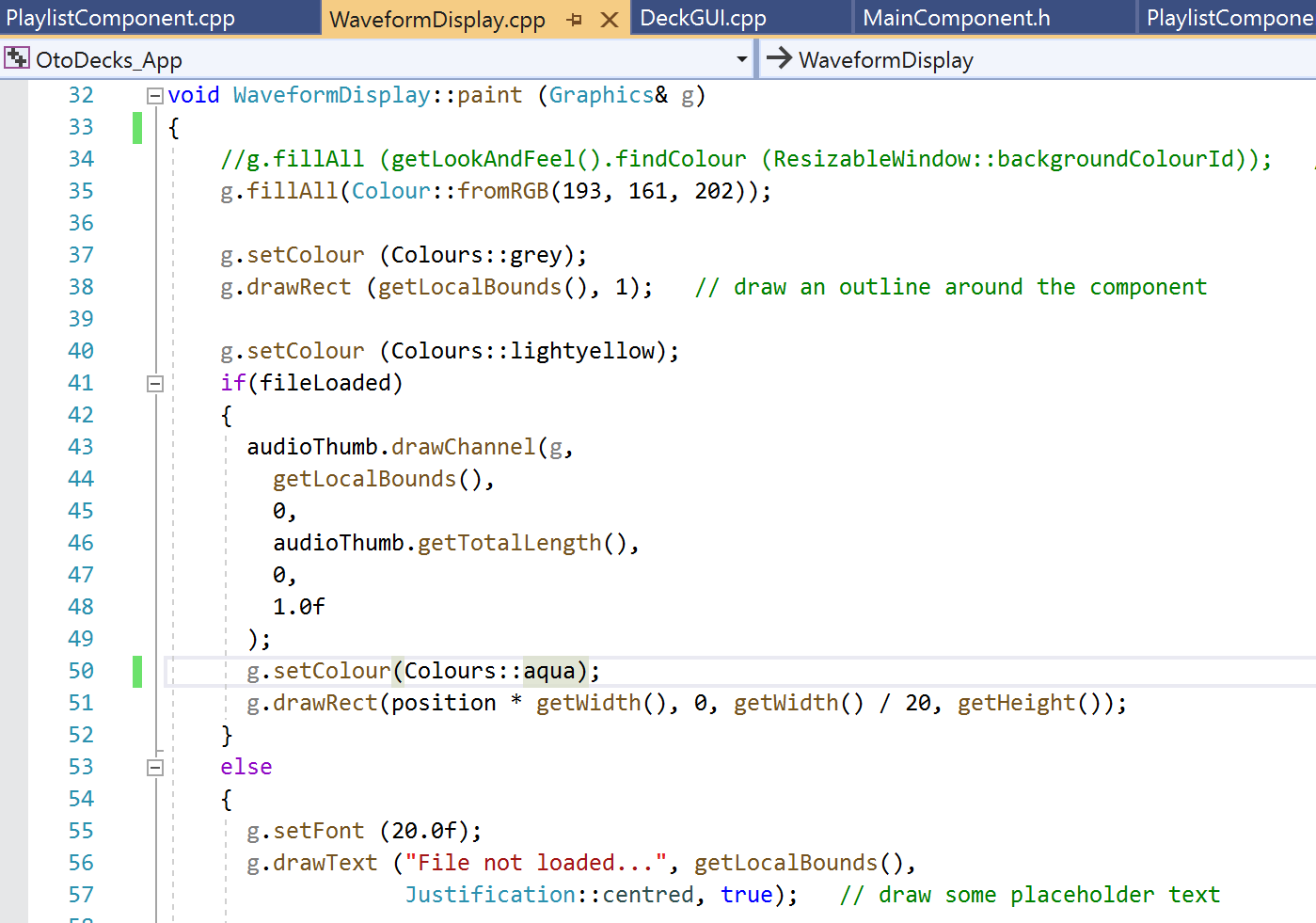


Figure 4.2 The paint function from waveformDisplay

The customization of playlist component can be seen in R3C, the highlight colour of selected row (figure 3.10) is set by RGB colour mode at line 86 of figure 4.3.

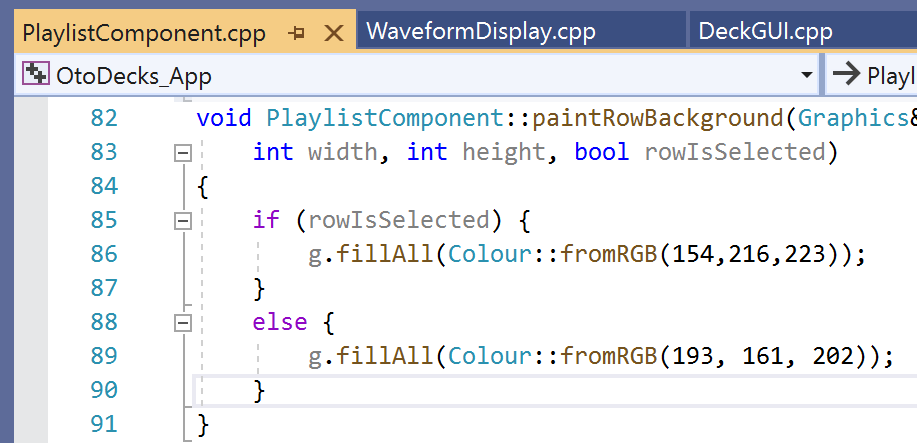


Figure 4.3 The paintRowBackground function from playlist Component

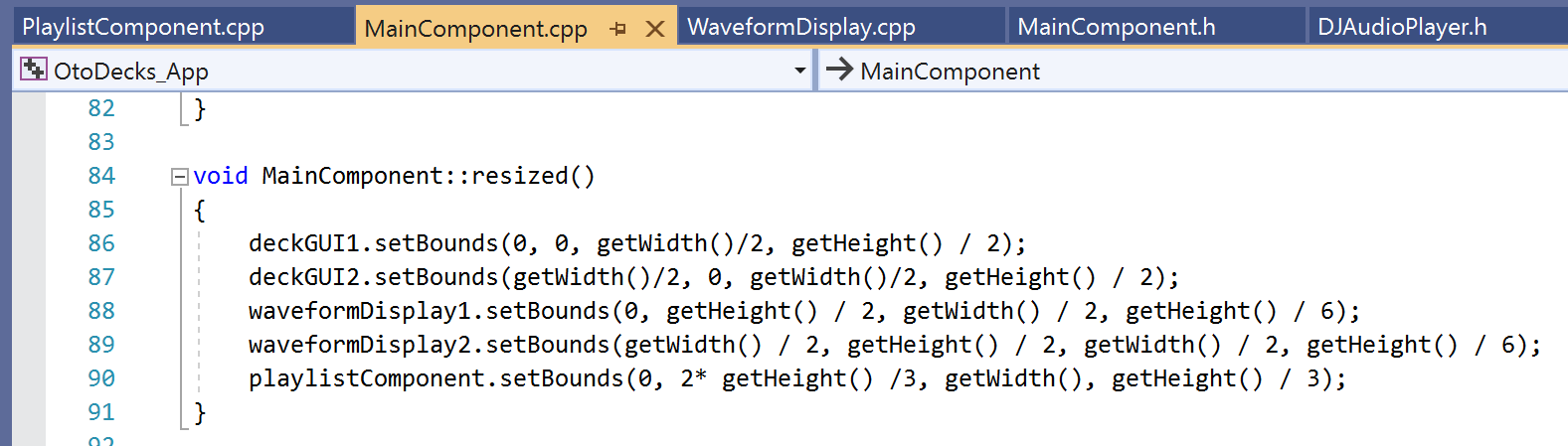


Figure 4.4 The resized function from MainComponent cpp

Two waveforms, two DeckGUIs and one playlist Components are included in the main Component cpp file. The setBounds function can set the size of each component which is with addAndMakeVisible() function in the constructor above.