Object Oriented Programming
Coursework for Endterm: Otodecks

Work by: Matthew Lim Joon Yan

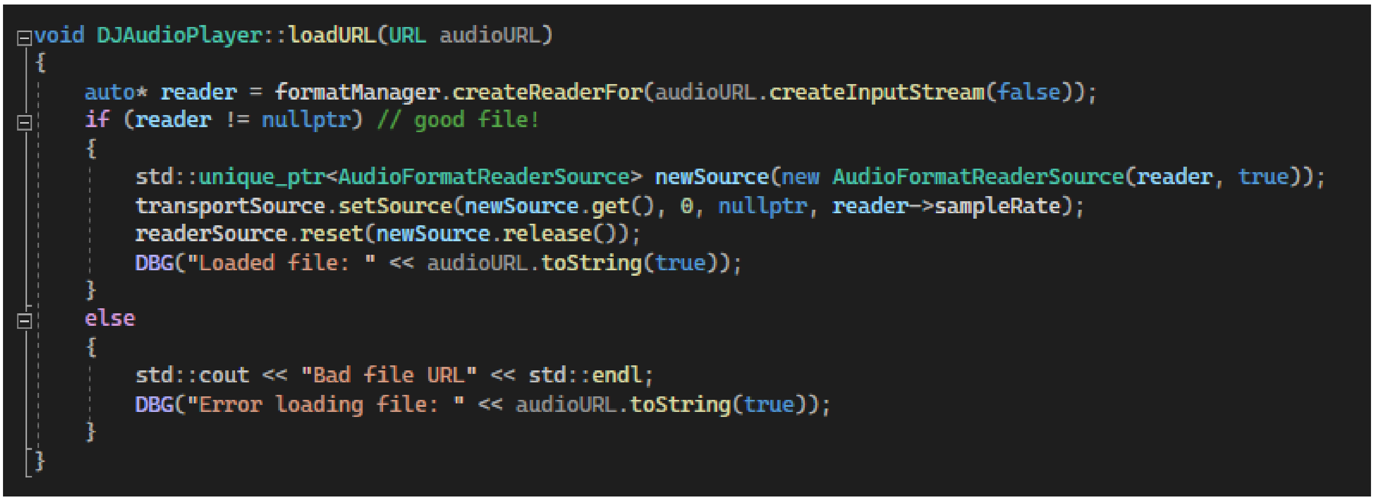
Introduction

Otodecks is a DJ application we were tasked to improve on as part of the final project in this course. This custom deck control component features original graphics and an inventive means of deck control. The music library component enables users to manage a music library within the application, search for music, and upload it to the decks. Meanwhile, enhancements have also been made to Otodecks by improving its functionality and user interface, resulting in a more versatile and user-friendly DJ application.

R1 – Basic Functionality

- R1A Loading audio files into the audio players

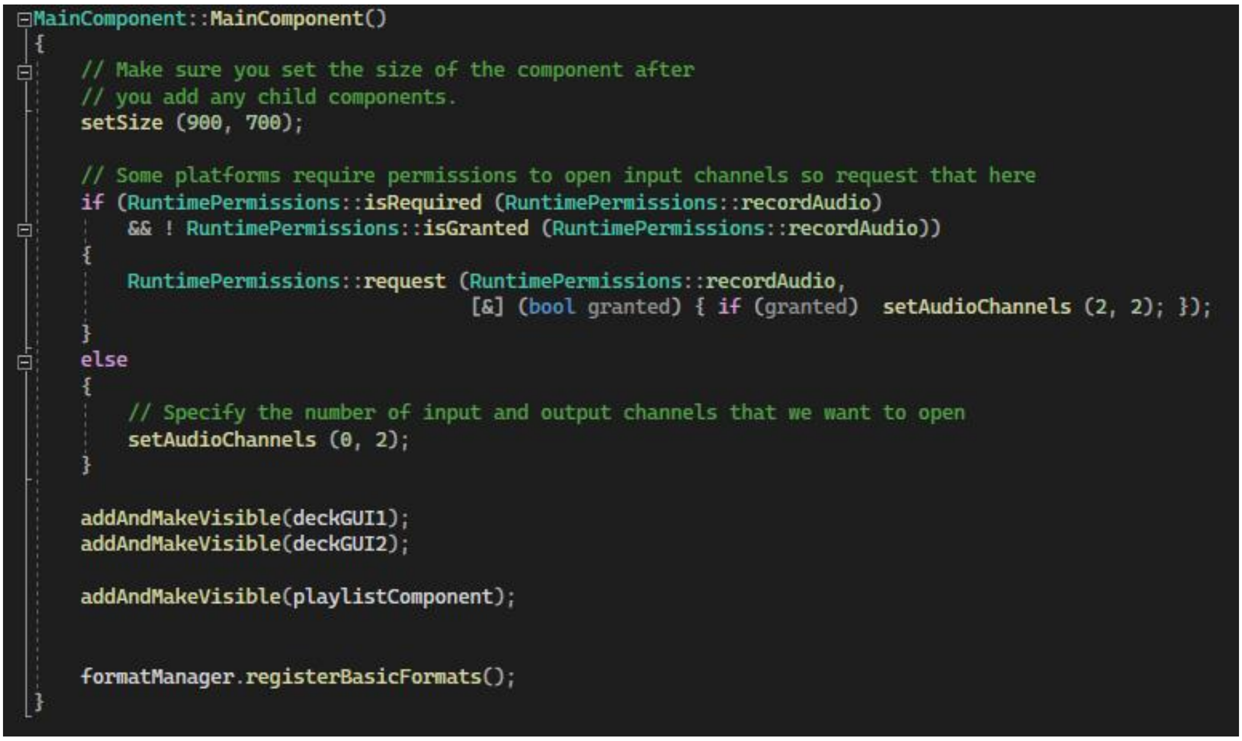
The function creates an AudioFormatReader object from the input stream using the createReaderFor method of the AudioFormatManager class. The process automatically detects the audio file format and makes a reader capable of reading it.



```
void DJAudioPlayer::loadURL(URL audioURL)
{
    auto* reader = formatManager.createReaderFor(audioURL.createInputStream(false));
    if (reader != nullptr) // good file!
    {
        std::unique_ptr<AudioFormatReaderSource> newSource(new AudioFormatReaderSource(reader, true));
        transportSource.setSource(newSource.get(), 0, nullptr, reader->sampleRate);
        readerSource.reset(newSource.release());
        DBG("Loaded file: " << audioURL.toString(true));
    }
    else
    {
        std::cout << "Bad file URL" << std::endl;
        DBG("Error loading file: " << audioURL.toString(true));
    }
}
```

- R1B Able to play tracks simultaneously

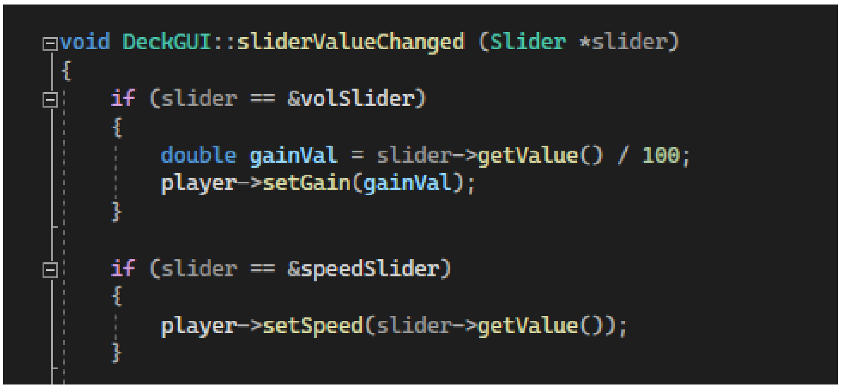
Two instances of the deck running simultaneously provide the user more functionality to control different audio separately.



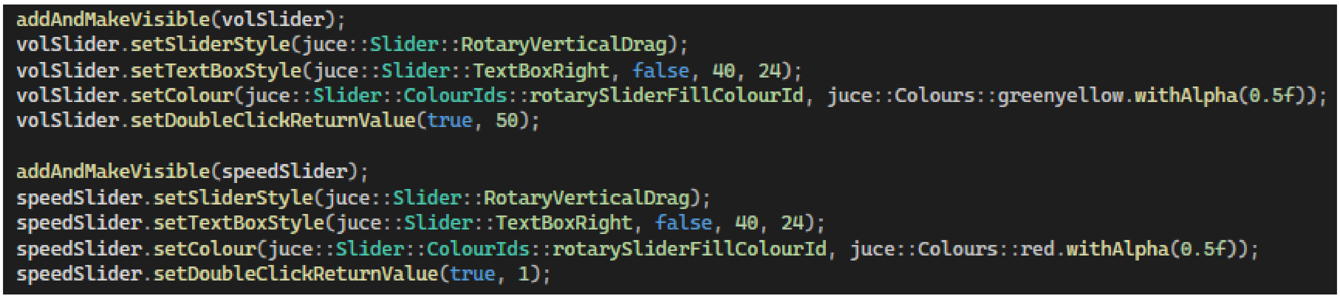
```cpp
MainComponent::MainComponent()
{
    // Make sure you set the size of the component after
    // you add any child components.
    setSize (900, 700);

    // Some platforms require permissions to open input channels so request that here
    if (RuntimePermissions::isRequired (RuntimePermissions::recordAudio)
        && ! RuntimePermissions::isGranted (RuntimePermissions::recordAudio))
    {
        RuntimePermissions::request (RuntimePermissions::recordAudio,
                                     [&] (bool granted) { if (granted)  setAudioChannels (2, 2); });
    }
    else
    {
        // Specify the number of input and output channels that we want to open
        setAudioChannels (0, 2);
    }

    addAndMakeVisible(deckGUI1);
    addAndMakeVisible(deckGUI2);

    addAndMakeVisible(playlistComponent);


    formatManager.registerBasicFormats();
}
```

- R1C Volume control / - R1D Track speed control

These custom knobs help control the track's volume and speed, enabling the user to adjust the audio to their liking.



```cpp
void DeckGUI::sliderValueChanged (Slider *slider)
{
    if (slider == &volSlider)
    {
        double gainVal = slider->getValue() / 100;
        player->setGain(gainVal);
    }

    if (slider == &speedSlider)
    {
        player->setSpeed(slider->getValue());
    }
```

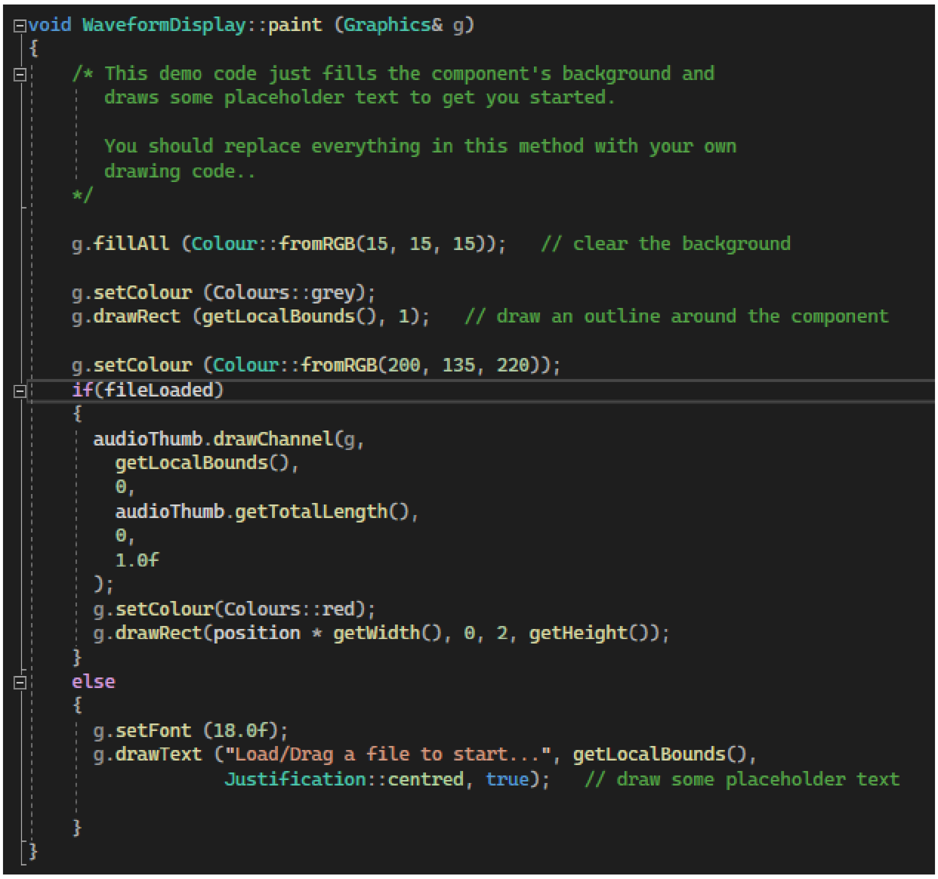


```cpp
addAndMakeVisible(volSlider);
volSlider.setSliderStyle(juce::Slider::RotaryVerticalDrag);
volSlider.setTextBoxStyle(juce::Slider::TextBoxRight, false, 40, 24);
volSlider.setColour(juce::Slider::ColourIds::rotarySliderFillColourId, juce::Colours::greenyellow.withAlpha(0.5f));
volSlider.setDoubleClickReturnValue(true, 50);

addAndMakeVisible(speedSlider);
speedSlider.setSliderStyle(juce::Slider::RotaryVerticalDrag);
speedSlider.setTextBoxStyle(juce::Slider::TextBoxRight, false, 40, 24);
speedSlider.setColour(juce::Slider::ColourIds::rotarySliderFillColourId, juce::Colours::red.withAlpha(0.5f));
speedSlider.setDoubleClickReturnValue(true, 1);
```

R2 – Deck Playback Control

- R2A Custom implemented graphics

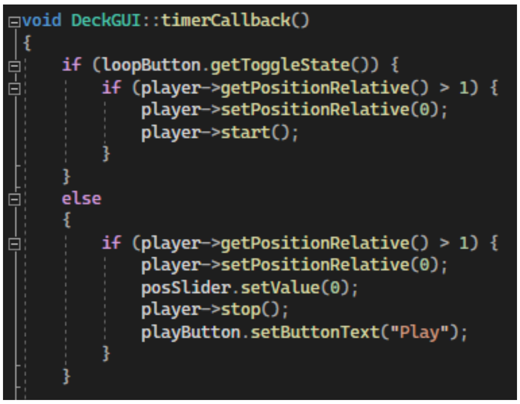
Changes were made to the look and feel of the user interface to make it look more welcoming and easier on the eyes. Sliders were evolved into knobs to better gauge what settings are in place for the current playing audio.



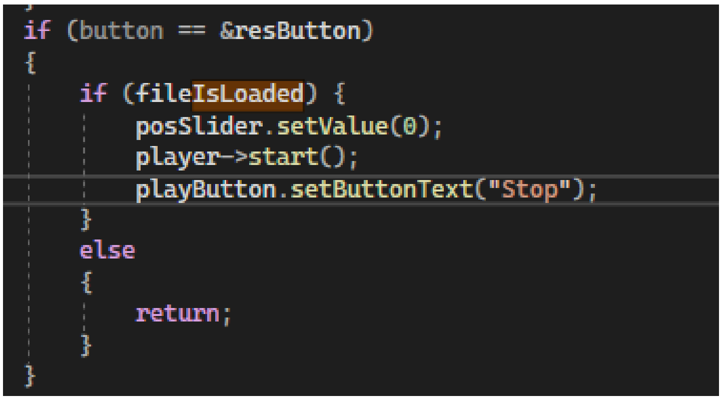
```
void WaveformDisplay::paint (Graphics& g)
{
    /* This demo code just fills the component's background and
       draws some placeholder text to get you started.

       You should replace everything in this method with your own
       drawing code..
    */

    g.fillAll (Colour::fromRGB(15, 15, 15));   // clear the background

    g.setColour (Colours::grey);
    g.drawRect (getLocalBounds(), 1);   // draw an outline around the component

    g.setColour (Colour::fromRGB(200, 135, 220));
    if(fileLoaded)
    {
      audioThumb.drawChannel(g,
        getLocalBounds(),
        0,
        audioThumb.getTotalLength(),
        0,
        1.0f
      );
      g.setColour(Colours::red);
      g.drawRect(position * getWidth(), 0, 2, getHeight());
    }
    else
    {
      g.setFont (18.0f);
      g.drawText ("Load/Drag a file to start...", getLocalBounds(),
                  Justification::centred, true);   // draw some placeholder text

    }
}
```

- R2B Playback control

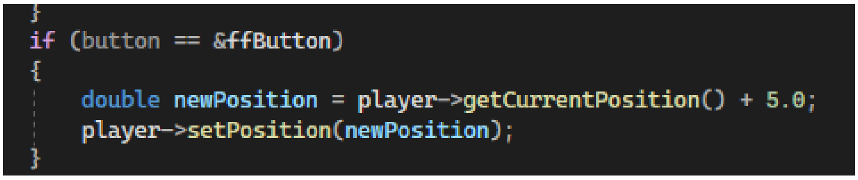
It included more functionality for the control of the audio. These buttons are a significant portion of the flexibility in management when using the improved Otodecks.



```
void DeckGUI::timerCallback()
{
    if (loopButton.getToggleState()) {
        if (player->getPositionRelative() > 1) {
            player->setPositionRelative(0);
            player->start();
        }
    }
    else
    {
        if (player->getPositionRelative() > 1) {
            player->setPositionRelative(0);
            posSlider.setValue(0);
            player->stop();
            playButton.setButtonText("Play");
        }
    }
```



```
if (button == &resButton)
{
    if (fileIsLoaded) {
        posSlider.setValue(0);
        player->start();
        playButton.setButtonText("Stop");
    }
    else
    {
        return;
    }
}
```

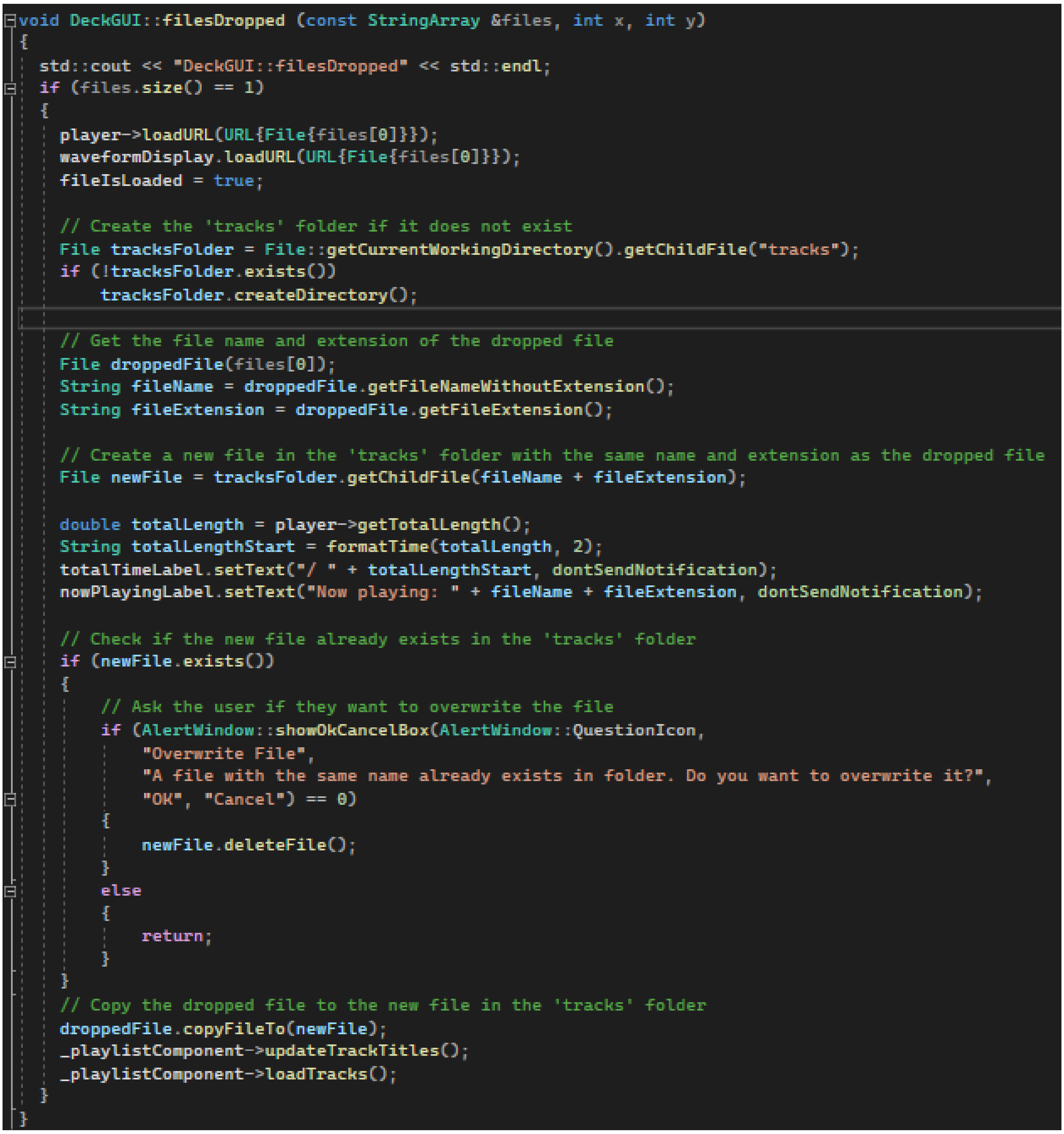


```
}
if (button == &ffButton)
{
    double newPosition = player->getCurrentPosition() + 5.0;
    player->setPosition(newPosition);
}
```

R3 – Music Library Component

- R3A Adding files to the music library

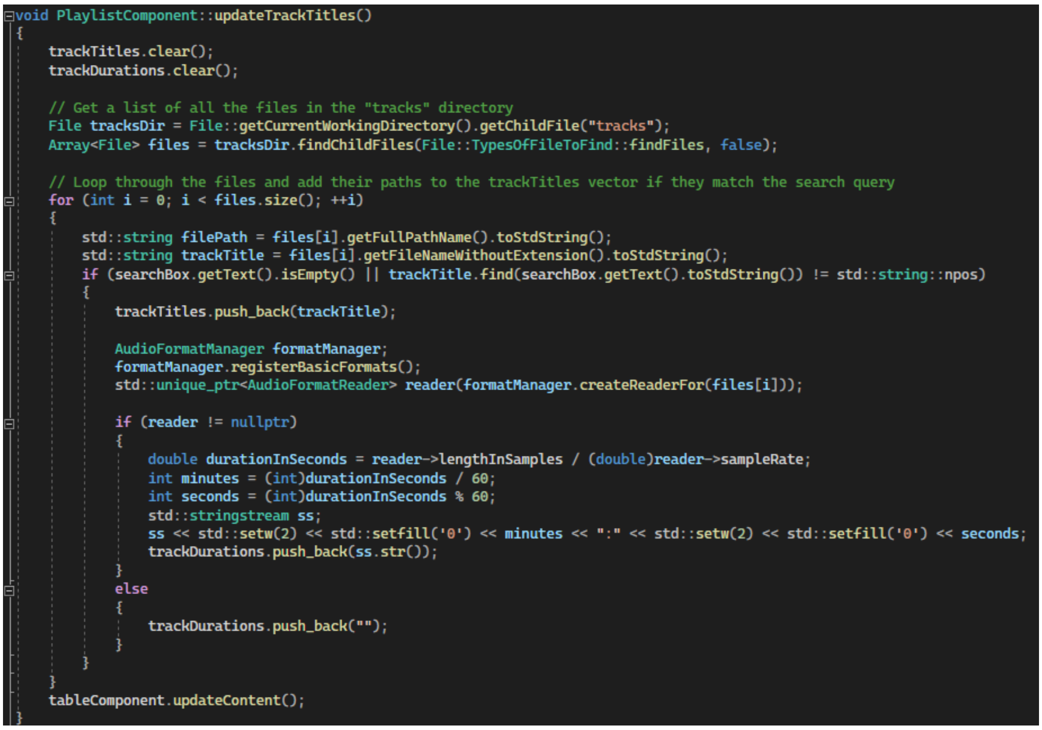
Significant changes were made to the adding system, where the focus was shifted to the drag and drop feature, allowing for adding to the music library. This helps with easy access to new music tracks and the freedom of playing them through a simple and familiar function.



```cpp
void DeckGUI::filesDropped (const StringArray &files, int x, int y)
{
    std::cout << "DeckGUI::filesDropped" << std::endl;
    if (files.size() == 1)
    {
        player->loadURL(URL{File{files[0]}});
        waveformDisplay.loadURL(URL{File{files[0]}});
        fileIsLoaded = true;

        // Create the 'tracks' folder if it does not exist
        File tracksFolder = File::getCurrentWorkingDirectory().getChildFile("tracks");
        if (!tracksFolder.exists())
            tracksFolder.createDirectory();

        // Get the file name and extension of the dropped file
        File droppedFile(files[0]);
        String fileName = droppedFile.getFileNameWithoutExtension();
        String fileExtension = droppedFile.getFileExtension();

        // Create a new file in the 'tracks' folder with the same name and extension as the dropped file
        File newFile = tracksFolder.getChildFile(fileName + fileExtension);

        double totalLength = player->getTotalLength();
        String totalLengthStart = formatTime(totalLength, 2);
        totalTimeLabel.setText("/ " + totalLengthStart, dontSendNotification);
        nowPlayingLabel.setText("Now playing: " + fileName + fileExtension, dontSendNotification);

        // Check if the new file already exists in the 'tracks' folder
        if (newFile.exists())
        {
            // Ask the user if they want to overwrite the file
            if (AlertWindow::showOkCancelBox(AlertWindow::QuestionIcon,
                "Overwrite File",
                "A file with the same name already exists in folder. Do you want to overwrite it?",
                "OK", "Cancel") == 0)
            {
                newFile.deleteFile();
            }
            else
            {
                return;
            }
        }
        // Copy the dropped file to the new file in the 'tracks' folder
        droppedFile.copyFileTo(newFile);
        _playlistComponent->updateTrackTitles();
        _playlistComponent->loadTracks();
    }
}
```

- R3B/R3C Displaying and searching song data in the library

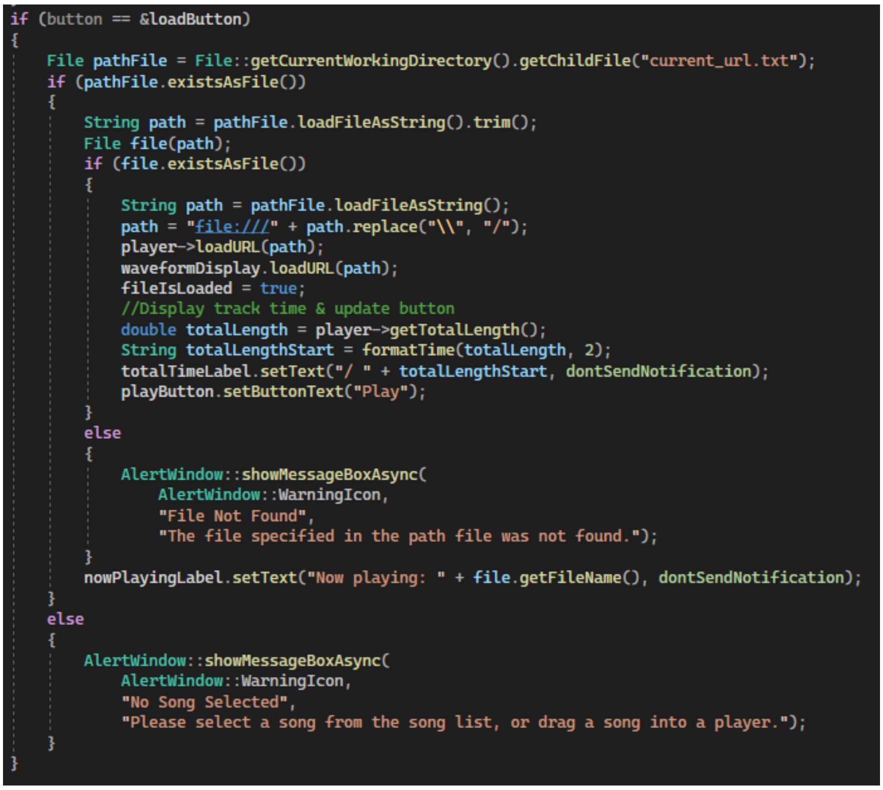
Song data is read and stored in vectors on startup, giving real-time updated information displayed to the user with ease of access. The search function also updates the information presented in real-time and quickly filters through the content in the existing library. This allows for quick finding and switching of tracks.



```cpp
void PlaylistComponent::updateTrackTitles()
{
    trackTitles.clear();
    trackDurations.clear();

    // Get a list of all the files in the "tracks" directory
    File tracksDir = File::getCurrentWorkingDirectory().getChildFile("tracks");
    Array<File> files = tracksDir.findChildFiles(File::TypesOfFileToFind::findFiles, false);

    // Loop through the files and add their paths to the trackTitles vector if they match the search query
    for (int i = 0; i < files.size(); ++i)
    {
        std::string filePath = files[i].getFullPathName().toStdString();
        std::string trackTitle = files[i].getFileNameWithoutExtension().toStdString();
        if (searchBox.getText().isEmpty() || trackTitle.find(searchBox.getText().toStdString()) != std::string::npos)
        {
            trackTitles.push_back(trackTitle);

            AudioFormatManager formatManager;
            formatManager.registerBasicFormats();
            std::unique_ptr<AudioFormatReader> reader(formatManager.createReaderFor(files[i]));

            if (reader != nullptr)
            {
                double durationInSeconds = reader->lengthInSamples / (double)reader->sampleRate;
                int minutes = (int)durationInSeconds / 60;
                int seconds = (int)durationInSeconds % 60;
                std::stringstream ss;
                ss << std::setw(2) << std::setfill('0') << minutes << ":" << std::setw(2) << std::setfill('0') << seconds;
                trackDurations.push_back(ss.str());
            }
            else
            {
                trackDurations.push_back("");
            }
        }
    }
    tableComponent.updateContent();
}
```

- R3D Load files from the library onto the deck

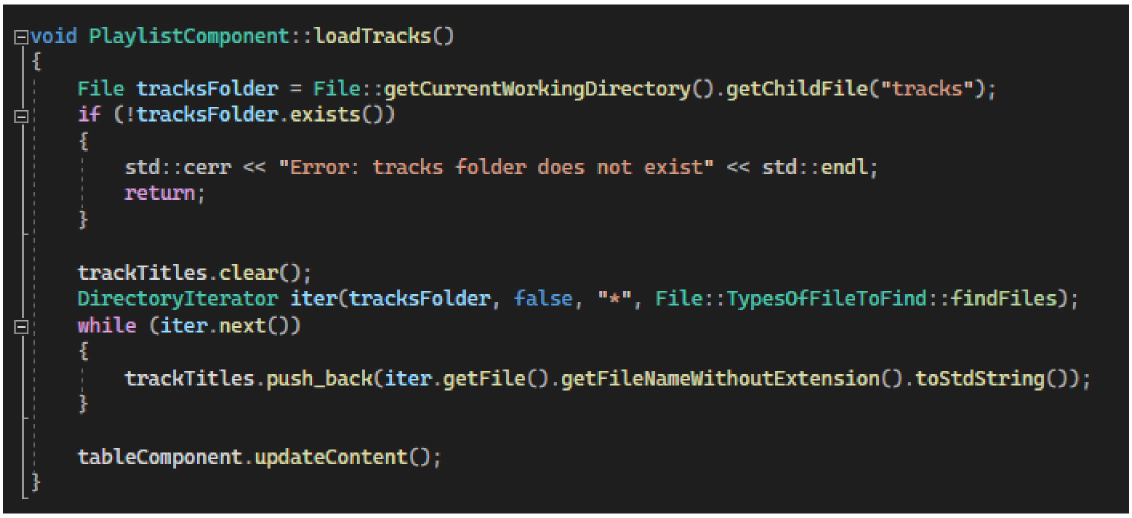
The “Load” button passes the URL of the selected file to the player, which loads the audio “.mp3” file. This indicates that a file is loaded, activates all the controls and syncs the audio and the visuals. Each deck has separate load buttons that load the selected song accordingly.



```cpp
if (button == &loadButton)
{
    File pathFile = File::getCurrentWorkingDirectory().getChildFile("current_url.txt");
    if (pathFile.existsAsFile())
    {
        String path = pathFile.loadFileAsString().trim();
        File file(path);
        if (file.existsAsFile())
        {
            String path = pathFile.loadFileAsString();
            path = "file:///" + path.replace("\\", "/");
            player->loadURL(path);
            waveformDisplay.loadURL(path);
            fileIsLoaded = true;
            //Display track time & update button
            double totalLength = player->getTotalLength();
            String totalLengthStart = formatTime(totalLength, 2);
            totalTimeLabel.setText("/ " + totalLengthStart, dontSendNotification);
            playButton.setButtonText("Play");
        }
        else
        {
            AlertWindow::showMessageBoxAsync(
                AlertWindow::WarningIcon,
                "File Not Found",
                "The file specified in the path file was not found.");
        }
        nowPlayingLabel.setText("Now playing: " + file.getFileName(), dontSendNotification);
    }
    else
    {
        AlertWindow::showMessageBoxAsync(
            AlertWindow::WarningIcon,
            "No Song Selected",
            "Please select a song from the song list, or drag a song into a player.");
    }
}
```

- R3E Music library persists

The library is updated in a folder and opened when the application boots up again. Any changes made to the library will be reflected and saved for the next session.



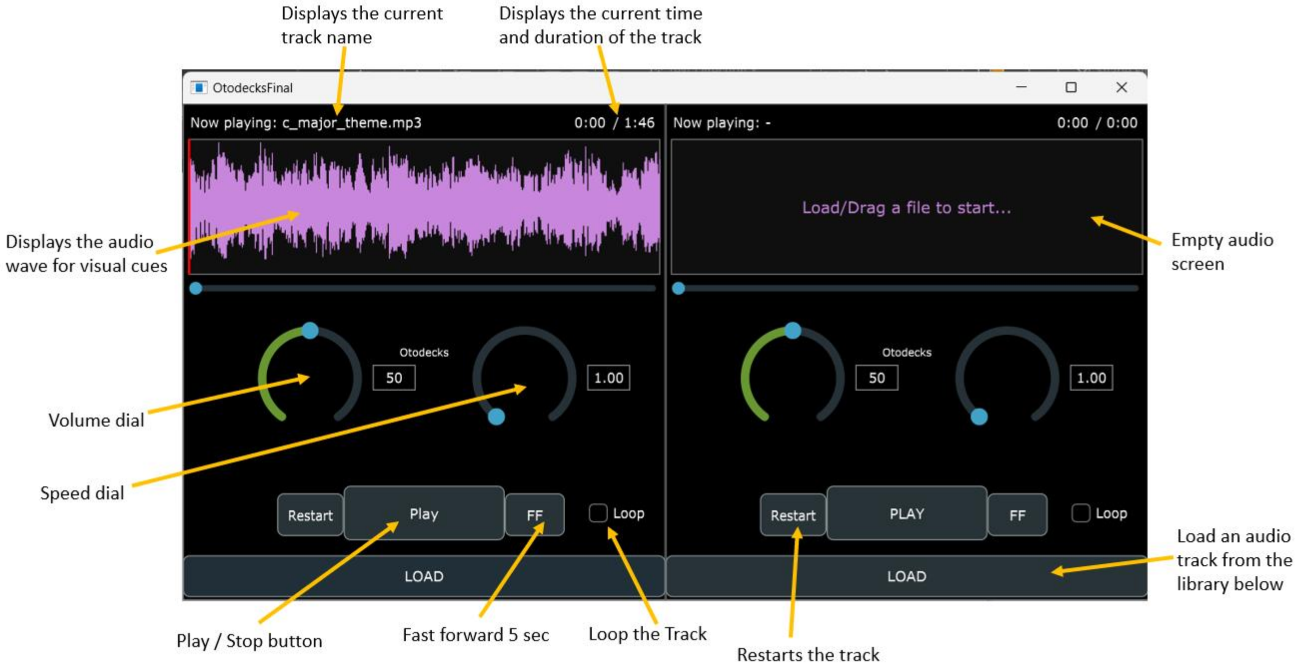
```
void PlaylistComponent::loadTracks()
{
    File tracksFolder = File::getCurrentWorkingDirectory().getChildFile("tracks");
    if (!tracksFolder.exists())
    {
        std::cerr << "Error: tracks folder does not exist" << std::endl;
        return;
    }

    trackTitles.clear();
    DirectoryIterator iter(tracksFolder, false, "*", File::TypesOfFileToFind::findFiles);
    while (iter.next())
    {
        trackTitles.push_back(iter.getFile().getFileNameWithoutExtension().toStdString());
    }

    tableComponent.updateContent();
}
```

R4 – Custom GUI

- R4A GUI Layout

Significant changes were made so that the user experience is better overall. Colour palettes are selected following a set theme so that all the colours blend well.

- R4B R2 Component



Displays the current track name

Displays the current time and duration of the track

Displays the audio wave for visual cues

Empty audio screen

Volume dial

Speed dial

Load an audio track from the library below

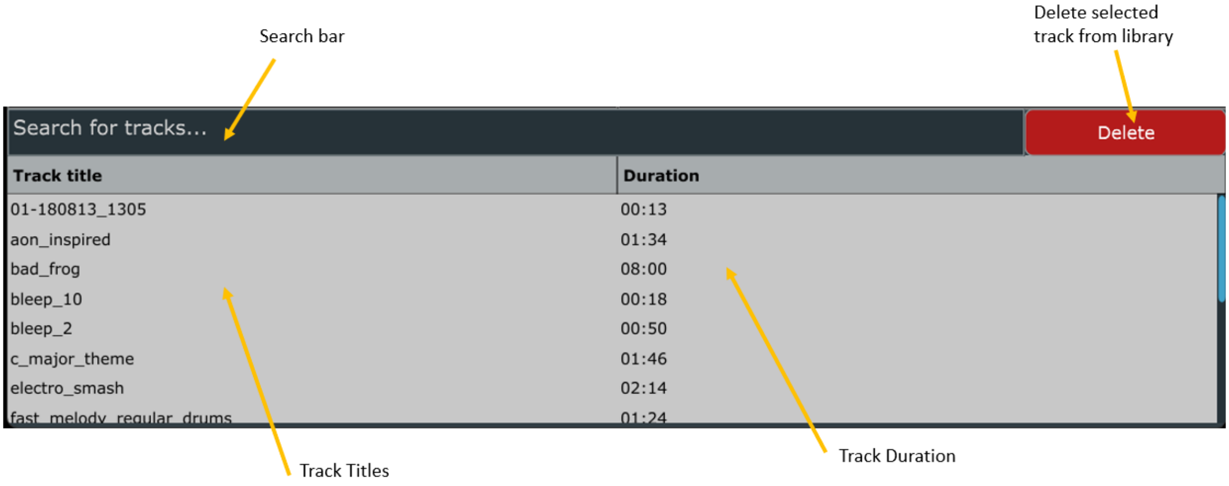
Play / Stop button

Fast forward 5 sec

Loop the Track

Restarts the track

- R4C R3 Component



Track title	Duration
01-180813_1305	00:13
aon_inspired	01:34
bad_frog	08:00
bleep_10	00:18
bleep_2	00:50
c_major_theme	01:46
electro_smash	02:14
fast_melody_regular_drums	01:24

Search bar

Delete selected track from library

Track Titles

Track Duration

Reflection

As a student who just completed a project on JUCE C++, I have gained valuable experience and knowledge. I have learned how to use the JUCE framework to create a functional and user-friendly GUI for my application. I also better understood the importance of proper project organisation, code optimisation, and debugging techniques. Working on this project has also helped me improve my problem-solving skills and taught me the importance of time management. Overall, this project has been an excellent learning opportunity and has given me a deeper appreciation for the intricacies of software development.