Object-oriented Programming Finals Report

Table of Contents

1. Introduction

The purpose of this report is to provide an overview of the audio application and to outline the process from conceptualisation to implementation.

In broad, I will be covering the following areas:

- Navigation of the application
- Module design
- Ideation and design of user interface
- New features (Playlist and Low/High Pass Filter)
- Ideas for future enhancements

The application is built in the Xcode development environment upon the JUCE framework.

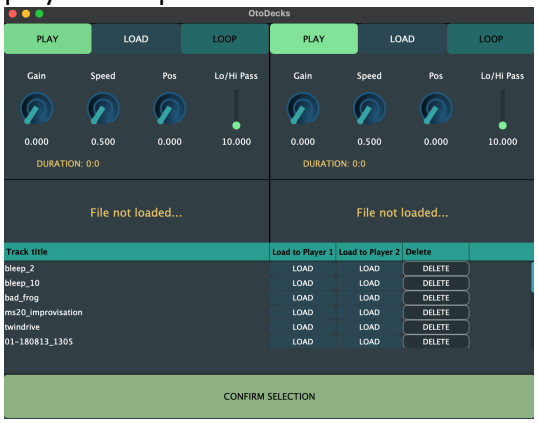
2. Overview of application

In this section, I will be covering the basic workflow a user goes through when using the application.

When the application first starts, it prompts the user to select a directory or a selection of audio files.

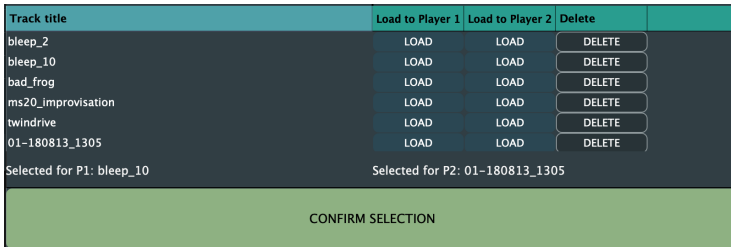
Screen Capture 1 Selecting files at start up

Once the tracks are selected, the audio application will open, with the tracks loaded into the playlist component.



Screen Capture 2 Overview of audio application

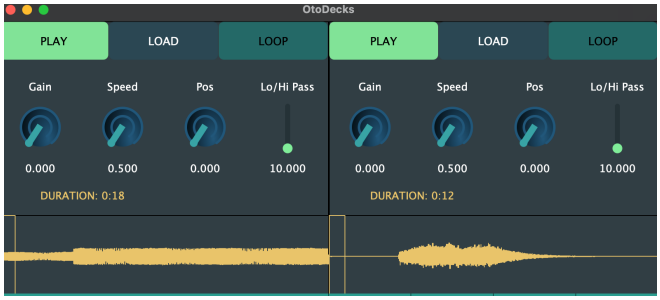
The application contains two audio players. The user can then click on the songs they want to load to each player. The songs they selected appears at the bottom of the playlist.



Screen Capture 3 Selected tracks

Once they are satisfied, they can confirm their selection by clicking the confirm button.

The songs will then be loaded onto the respective players. The user will receive visual confirmation when the waveform of the audio file is drawn.



Screen Capture 4 Waveform drawn

The user will have the choice of playing the song one at a time or both concurrently. They can adjust the volume, position of the track, speed of the track and apply a high pass filter on the track.

3. Basic Program

In this section, I will be covering the modules that makes up the basic functionality of the application (R1).

The OtoDecksApplication uses the JUCE framework. JUCE is an open source codebase written in C++ that allows us to produce an audio application that works across different operating systems. (Raw Material Software Limited, 2024)

The OtoDecksApplication inherits from the JUCEApplication module. The main() routine launches the application.

The application has seven modules :

1. **MainComponent**
    - Prepares and processes audio from the two audio players.
    - Serves as a button listener for the 'confirm selection' button, where users load two tracks from the playlist to the player
2. **DJAudioPlayer**
    - Prepares resources to play audio tracks
    - Controls the volume, speed and position
    - Controls if the track is looped (Requirments 3)
    - Applies the IIR low pass and high pass filters (Requirements 3)
3. **DeckGUI**
    - Interface between the sliders and buttons and the DJAudioPlayer
    - In charge of loading files onto the audio player
    - Instantiates and draws the WaveformDisplay object
    - Performs arrangement of child components (Requirements 2)

4. **WaveformDisplay:** Uses the audio thumbnail to show a preview of the file

5. **CustomStyle (Requirements 2)**
    - Custom Rotary Slider
    - Custom Vertical Slider

- Custom play, load, delete, loop and confirm text buttons

6. **Track (Requirements 3)**
    - Track object contains a type_name that indicates if the track is to be loaded to player1 or player2.
    - Track object contains a URL which is the path to the track chosen.

7. **PlaylistComponent (Requirements 3)**
    - Prompts user to load audio tracks at the launch of the application
    - Provides user with an overview of audio track titles
    - Allows user to load a specific track to a specific player
    - Allows user to delete tracks from the playlist

4. Design and ideation of user interface

4.1 Research on user interface

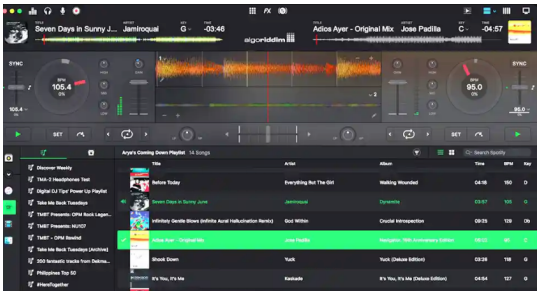
My research on DJ audio applications' user interface design focuses on two parts:

- How components are normally laid out
- Conventional colour schemes used in DJ application

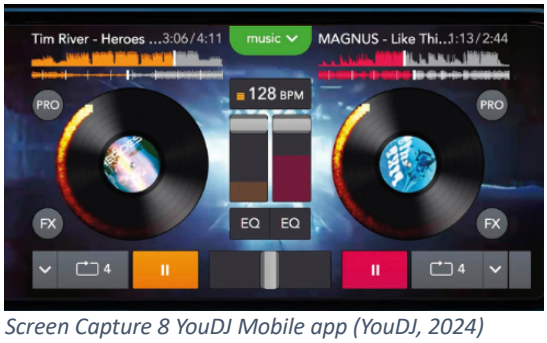
I gained inspiration from applications such as rekordbox, Serato, DJay Pro, YouDJMixer. (Feasey-Kemp, 2023)

Screen Capture 5 rekordbox by Pioneer DJ

Screen Capture 6 Serato DJ



Screen Capture 7 DJay Pro



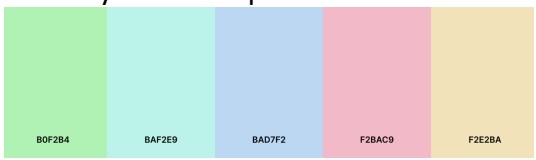
Screen Capture 8 YouDJ Mobile app (YouDJ, 2024)

These software share some commonalities in terms of its design – the audio thumbnails are placed at the top and the playlist component at the bottom. The colour schemes are dark, with white text indicating what each component does. All of the software allow users to

import and manage tracks. They allow tracks to be looped and for the position of the tracks to be altered by the user. Most of the applications also allows users to apply filters on the music track. I intend to do something similar, with the two DeckGUI components at the top and the playlist component on the bottom.

4.2 Colour schemes and initial mock-up

I initially wanted a pastel colour scheme for the DJ application, such as the palettes below.



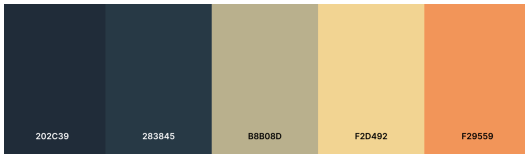
Screen Capture 9 Pastel Selection 1

Screen Capture 10 Pastel Selection 2

However, my research revealed that DJ applications models after physical DJ stations in clubs and concerts, which typically have dark settings. I decided that a darker user interface would be more appropriate.

I decided to have a dark background, with accents from the palettes below.

Screen Capture 11 Accent and highlights 1



Screen Capture 12 Accent and highlights 2

I also made a mock-up of what my application should look like.

Screen Capture 13 Mockup done by me with free elements on Canva

4.3 CustomStyle class

I created a CustomStyle class to decouple the design of buttons and sliders from the main modules. This class inherits from Juce's LookAndFeel_V4 class. For example, the 'load button' is used in both the deckGUI and the playlist component and has the same style. Hence, creating a separate class and a custom button will allow me to reuse this button in different parts of the program.

The CustomStyle class contains custom buttons and sliders that inherits from JUCE's original buttons and sliders. Buttons and sliders have an enumeration 'ColourIds' that allow developers to change the colours of the components. I made use of this to change the components and fit the colour scheme.

```
LoadButton::LoadButton()
{
    /** button have no outline */
    setColour(ComboBox::outlineColourId, juce::Colours::transparentBlack);
    /** button cannot toggle*/
    setClickingTogglesState(false);
    setButtonText("LOAD"); // the button text
    setColour(buttonColourId, juce::Colour(38, 70, 83)); // set button to be blue
    setColour(textColourOnId, juce::Colours::white); // set text white
}
```

Screen Capture 14 Example of custom button component

The buttons have a transparent outline to create a clean and sophisticated look. (Debian, 2020)

4.4 FlexBox and Drawable text

4.4.1 FlexBox

In order to place the components in the deckGUI as per the mock-up above, I made use of the FlexBox class to manage the layout of the content.

Each FlexBox container contains and manages the layout of a set of FlexItem objects. (JUCE, 2023) It allows us to specify the direction of the main axis and how the items are spaced in the container.

For example, I have defined a new FlexBox object "flexbox_buttons".

```
// First section contains buttons and takes up 1 row
FlexBox flexbox_buttons;
flexbox_buttons.flexDirection = FlexBox::Direction::row;
flexbox_buttons.flexWrap = FlexBox::Wrap::wrap;
flexbox_buttons.alignContent = FlexBox::AlignContent::spaceAround;
```

Screen Capture 15 FlexBox flexbox_buttons

I added the play, load and loop buttons as FlexItems to an array of FlexItems. I then added this array to the flexbox_buttons FlexBox container.

```
// JUCE array object acts like a vector
Array<FlexItem> play_load_loop_buttons;

/** Each button takes up 1/3 of the width of the row */
play_load_loop_buttons.add(FlexItem(getWidth()/3,rowH,playButton));
play_load_loop_buttons.add(FlexItem(getWidth()/3,rowH,loadButton));
play_load_loop_buttons.add(FlexItem(getWidth()/3,rowH,loopButton));

flexbox_buttons.items = play_load_loop_buttons;
```

Screen Capture 16 Array of FlexItems

The performLayout method of the FlexBox container allows me to position the buttons in the first row.

```
/** buttons takes up first row, returns the bottom of that */
flexbox_buttons.performLayout(bounds.removeFromTop(rowH));
```

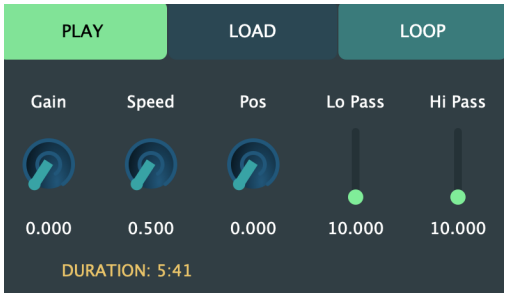
Screen Capture 17 Perform Layout method on FlexBox

Using FlexBox and flex items allows for separation of items into different categories.

In addition, the performLayout method, combined with the removeFromTop() Rectangle method allows me to position the components without explicitly setting hard-coded coordinates. If I decide to alter the size of the application down the road, I will only need to change the size in the main component and the proportions of the components will remain the same.

4.4.2 DrawableText

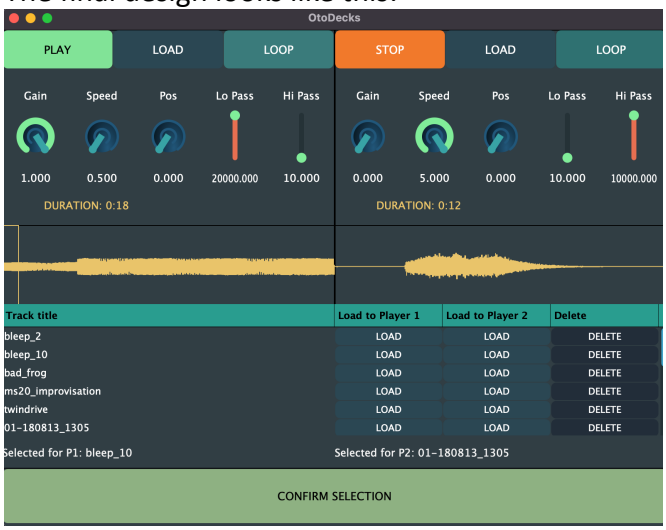
In addition, I used DrawableText in two occasions – to display a selected track's title at the bottom of the playlist, and to display the duration of a track within the deckGUI. (Raketa, 2014) The drawable text is positioned with Juce's Rectangle class.



Screen Capture 18 Example of DrawableText: duration shown at the bottom of sliders

4.5 Final Design

The final design looks like this:



Screen Capture 19 Final design of the OtoDecks application

5. New Features

To find new features to implement, I did some research on what users would want from a DJ application. DJ software is used to create mixes and hence, would require the capacity to do the following things (Feasey-Kemp, 2023):

- Pause and play a track
- Play multiple tracks concurrently
- Change the speed of playback
- Providing an easy-to-access library for users to select tracks
- Allow for the user to create 'sets'
- Provide a way to apply filter on a track
- Crossfade between tracks
- Beat matching between two tracks

The first three basic functionalities (playing a track, playing multiple tracks, changing speed) have been satisfied with the fulfilment of Requirement 1 of the project. I have also combined the play and stop button into one single button, that toggles between playing the track and stopping the audio stream.

Among the many functions of an audio application, I choose to focus on creating a viable playlist for the user, and two filters to attenuate high or low frequency signals.

5.1 Playlist Component

A playlist is especially integral for an DJ application. Most DJs would prepare a list of tracks before the event. They would then choose the tracks onsite on-the-fly from the planned playlist, based on the vibe and energy of the audience. (Sokolovskiy, 2024) Hence, I chose to implement a playlist to make the Otodecks application more robust.

In order to load a track from the playlist selection, I created the Track class. A Track object will hold two attributes:

1. Type_name (string) indicates the identity of the track – whether it should be loaded to player 1, player 2, or serves as a spare track.
2. Track_url (URL) is the file path to the select audio track.

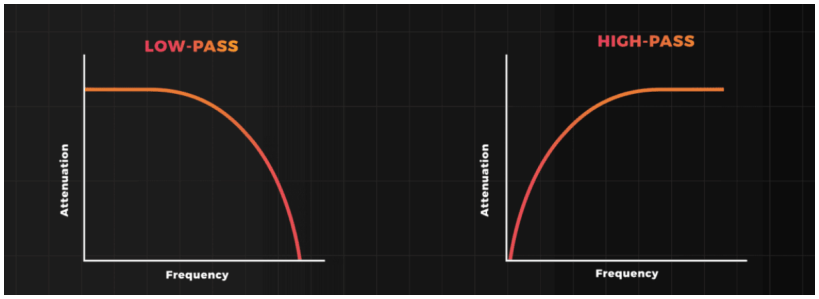
The user is prompted to select audio tracks at the start of the application. These tracks will be loaded on the playlist, with the track titles in the first column. Three Track objects are created – the track to be loaded to player 1, the track to be loaded to player 2 and the spare back-up track.

The user can then click the load button to select the track for a particular player. The load button will alter the url attribute of the respective Track objects. The title of the selected track can be seen at the bottom of the playlist component. The track can also be deleted with the buttons in the delete column.

Once the user has completed selection, they can then confirm the selection. This event is registered by the main component, which will get the two player 1 and player 2 tracks from the playlist component.

5.2 Low and High pass filter

I have chosen to work on filters for my extension as they are important for DJs to alter the tone colour of a sound, to erase noise and finetune certain parts of a track. I have decided to focus on the low pass and high pass filters.



Screen Capture 20 Low and high Pass filters on attenuation vs frequency (MGF Audio, 2023)

A low pass filter is a frequency domain filter that attenuates high-frequency components and preserves the low-frequency components. (Sanchhaya Education Private Limited, 2023) It functions by only passing signals that have a frequency that is lower than a selected cutoff frequency. (Analog Devices, 2024)

A low pass filter can be used to add clarity to tracks by removing unwanted frequencies, such as the 'shrills' in a badly recorded track. (Messitte, 2023)

Screen Capture 21 Highlighted blue part will be removed in low pass filters (Miraglia, 2023)

Working in reverse, a high pass filter is a frequency domain filter that amplifies high-frequency components and suppresses low-frequency ones. A high pass filter can be used to remove unwanted noise at low frequencies which may be unwittingly reproduced by high-end speakers. (Messitte, 2022)

Screen Capture 22 Highlighted red part will be removed in high pass filters (Miraglia, 2023)

I implemented the filters using Juce's IIR Audio Sources, which performs an IIR filter on another audio source.

```
void DJAudioPlayer::setLowPass(double hertz)
{
    if(isPlaying) // necessary because reader is only created by format manager in loadURL
    {
        auto* reader = readerSource -> getAudioFormatReader();

        // low pass filter attenuates frequencies below cutoff frequencies
        low_source.setCoefficients(IIRCoefficients::makeLowPass(reader->sampleRate, hertz));
    }
}

void DJAudioPlayer::setHighPass(double hertz)
{
    if(isPlaying) // necessary because reader is only created by format manager in loadURL
    {
        auto* reader = readerSource -> getAudioFormatReader();

        // high pass filter attenuates frequencies above cutoff frequencies
        high_source.setCoefficients(IIRCoefficients::makeHighPass(reader->sampleRate, hertz));
    }
}
```

Screen Capture 23 Low pass and high pass implementation in DJAudioPlayer

The user can choose the cut-off frequency for the low pass and high pass sliders using the two vertical sliders. The values given by the user will be used to set the coefficients of the IIR audio source.

```
// audio is passed serially transport -> low -> high -> resample
IIRFilterAudioSource low_source{&transportSource, false}; // attenuates low frequencies
IIRFilterAudioSource high_source{&low_source, false}; // attenuates high frequencies
// audio source that takes in an input and changes the sample rate
ResamplingAudioSource resampleSource{&high_source, false, 2};
// End of new code
```

Screen Capture 24 IIRFilterAudioSource declared in DJAudioPlayer's header file

The audio sources are then passed serially into the player's resampled source.

6. Conclusion and future enhancements

The OtoDecks application in this final project allows the user to load tracks from a playlist and play two concurrently. It also provides basic features such as an audio thumbnail and low and high pass filters. The modular nature of the project grants the ability for future modification to include more tracks playing concurrently at one.

Should I continue to work on the project in the future, I will implement the following improvements:

- A customizable user interface where users can choose their colour themes
- Implement equalization, reverb and cross-fading
- Make accessibility features such as navigation with key presses
- Provide function to record and save their mixes

In conclusion, making the OtoDecks application has given me a deeper appreciation for object-oriented techniques and modular designs. The principle of inheritance has allowed me to create the CustomStyle class without re-inventing the wheel and polymorphism has allowed me to control the interaction between modules. Object-oriented programming has also allowed me to create a project that is scalable with its reusable modules.

Bibliography

YouDJ, 2024. *YouDJ.* [Online]
Available at: https://you.dj/free-dj-software
[Accessed 2 November 2023].

Feasey-Kemp, N., 2023. *The 7 Best DJ Mixing Apps.* [Online]
Available at: https://dj.studio/blog/apps-for-dj-mixing
[Accessed 1 November 2023].

Raw Material Software Limited, 2024. *JUCE.* [Online]
Available at: juce.com
[Accessed 5 January 2024].

Feasey-Kemp, N., 2023. *How does DJ software work? The basics explained.* [Online]
Available at: https://dj.studio/blog/how-dj-software-works
[Accessed 10 January 2024].

Sokolovskiy, D., 2024. *How I prepare my DJ playlists.* [Online]
Available at: https://dsokolovskiy.com/blog/all/my-dj-playlists/
[Accessed 5 January 2024].

Sanchhaya Education Private Limited, 2023. *Difference between Low pass filter and High pass filter.* [Online]
Available at: https://www.geeksforgeeks.org/difference-between-low-pass-filter-and-high-pass-filter/
[Accessed 15 December 2023].

Analog Devices, 2024. *Low-Pass Filter.* [Online]
Available at: https://www.analog.com/en/resources/glossary/low-pass-filter.html
[Accessed 2 December 2023].

Messitte, N., 2023. *6 Ways to Use a Low Pass Filter When Mixing.* [Online]
Available at: https://www.izotope.com/en/learn/6-ways-to-use-a-low-pass-filter-when-mixing.html#:~:text=A%20low%2Dpass%20filter%20(also,of)%20higher%2Dfrequency%20signals.
[Accessed 5 January 2024].

Debian, 2020. *How to set the colour of the outline on a textbutton?.* [Online]
Available at: https://forum.juce.com/t/how-to-set-the-colour-of-the-outline-on-a-textbutton/37925/3
[Accessed 25 December 2023].

Raketa, 2014. *Resize Drawable Text.* [Online]
Available at: https://forum.juce.com/t/resize-drawabletext/13912
[Accessed 25 December 2023].

JUCE, 2023. *FlexBox Class Reference.* [Online]
Available at: https://docs.juce.com/master/classFlexBox.html
[Accessed 25 December 2023].

Messitte, N., 2022. *6 Ways to Use a High Pass Filter When Mixing.* [Online]
Available at: https://www.izotope.com/en/learn/6-ways-to-use-a-high-pass-filter-when-mixing.html#:~:text=A%20high%2Dpass%20filter%20is,of)%20lower%2Dfrequency%20signals.
[Accessed 14 February 2024].

Miraglia, D., 2023. *Unlock The Magic Of Low Pass Filters: The Definitive Guide To Transforming Your Mixes.* [Online]

Available at: https://unison.audio/low-pass-filters/
[Accessed 14 February 2024].
MGF Audio, 2023. *Audio Filter Types.* [Online]
Available at: https://producerhive.com/music-production-recording-tips/audio-filter-types/
[Accessed 14 February 2024].