**Report: DJ application enhancements**

**Introduction**

This document outlines the improvements made to the DJ application. The user interface has been redesigned, and a deck for sample management has been added. The play/stop control buttons have been removed. The playlist functionality has been expanded, allowing tracks to be loaded faster directly from the root folder.

**Core functionality (R1)**

Description of implemented features

**R1A:** Loading audio files into players

Tracks are loaded using the loadTrack method in the DeckGUI class. This method calls player->loadURL(audioURL) to load the track into DJAudioPlayer and waveformDisplay.loadURL(audioURL) to display the waveform.

**R1B:** Playing multiple tracks simultaneously

Track playback is managed by the playButton. When pressed, the buttonClicked method checks whether a track is currently playing and calls player->start() to start playback or player->stop() to stop it. The button state is updated using the playButtonSetColor method.

**R1C:** Mixing tracks by adjusting volume levels

Volume control is implemented using the volSlider. When the slider value changes, the sliderValueChanged method calls player->setGain(volSlider.getValue()) to adjust the volume.

**R1D:** Adjusting playback speed

Playback speed is controlled by the speedSlider. The sliderValueChanged method calls player->setSpeed(speedSlider.getValue()) to set the new speed.

**User interface customization (R2)**

The interface has undergone significant changes:

* Sliders have been changed from horizontal to vertical orientation.
* A file tree has been added to the left of the playlist.
* The playlist allows batch loading of all tracks from a folder with one click.
* Users can choose which player to load a track into.
* Track position can be adjusted using the waveform display.
* The deck layout, which displays track playback, can be rotated.
* The Load button has been removed.
* A Play/Stop button has been added.
* Each slider is now labeled.
* Minor changes have been made to the color scheme of the waveform display and buttons.

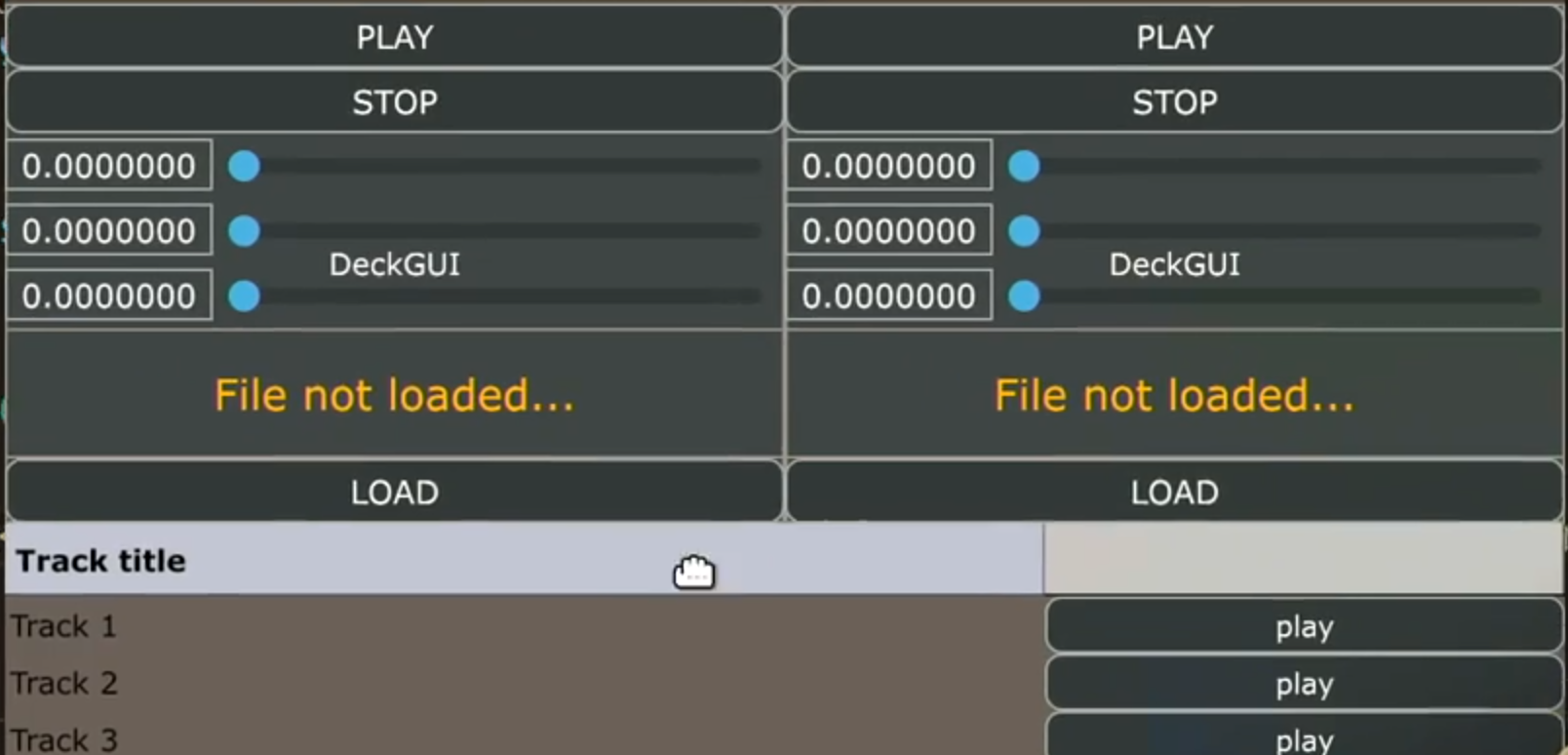


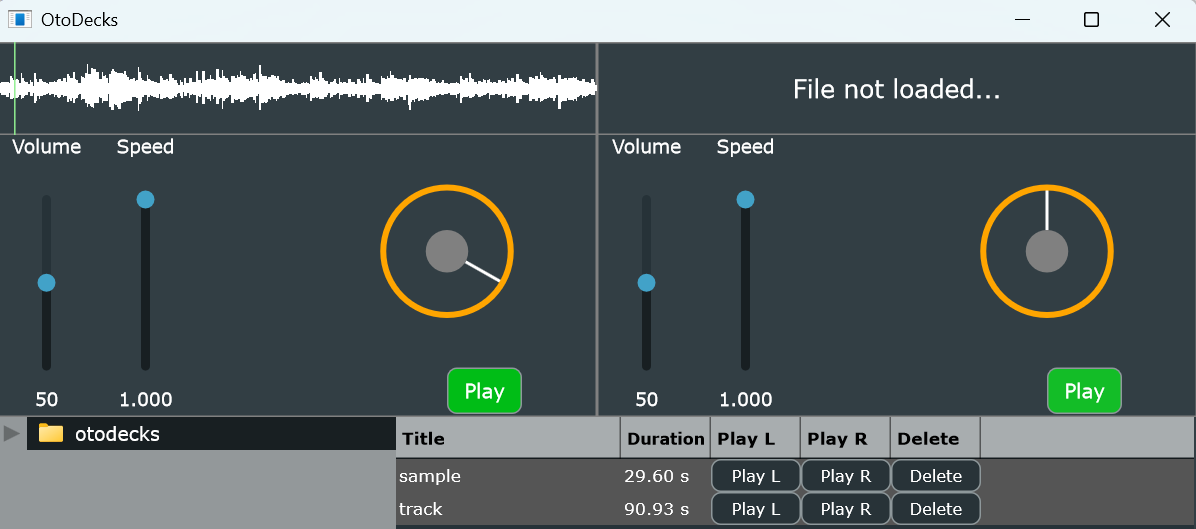
Image 1. The app before the changes  


Image 2. The app after all the changes.

**New feature implementation (R3)**

**Research**

The inspiration for new features came from observing [VirtualDJ](https://www.youtube.com/watch?v=GIPHwnokWiw&t=1149s). The simplicity and functionality of the software influenced the decision to implement deck management and a track-loading tree.



Image 3. General view of the virtual DJ application.

How it looks in my app.

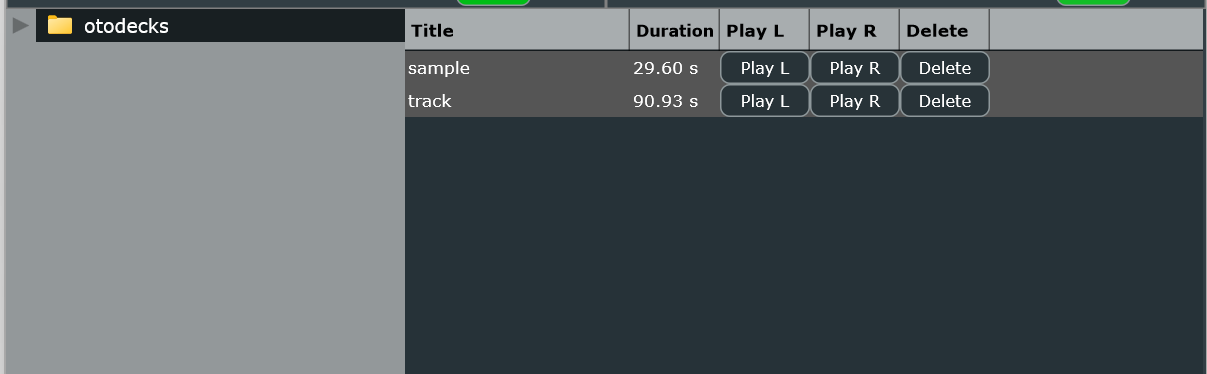


Image 4. Part of the OtoDecks interface.

Other DJ applications showcasing similar features were also reviewed ([video reference](https://www.youtube.com/watch?v=gaVJ-8qjulI&t=1s)).



Image 5. Screenshot from YouTube



Image 6. Screenshot from YouTube

Additionally, examples from JUCE's Projucer were examined for implementation insights.

**Code overview**

The application consists of multiple key components:

* MainComponent: handles the overall layout and initializes audio processing.
* DJAudioPlayer: manages track loading, playback, speed control, and volume control.
* DeckGUI: provides the user interface for each DJ deck, including waveform display and controls.
* PlaylistComponent: manages the playlist, allowing tracks to be loaded into decks.
* FileBrowserComponent: enables browsing and adding tracks from directories.
* InfiniteRotarySliderComponent: implements an infinite rotary slider for precise track position control.
* WaveformDisplay: displays the waveform of the loaded track and enables seeking.

**Key functionalities**

* MainComponent initializes the DeckGUI, PlaylistComponent, and FileBrowserComponent.
* DJAudioPlayer uses juce::AudioTransportSource and juce::ResamplingAudioSource for playback and speed adjustment.
* DeckGUI handles user interactions and updates UI components such as sliders and buttons.
* PlaylistComponent manages track metadata and interaction with decks.
* FileBrowserComponent facilitates track loading from folders.
* InfiniteRotarySliderComponent provides track scrubbing functionality.

Implementation of file browser and playlist components

**FileBrowserComponent**

**Used classes:**

* juce::FileTreeComponent
* juce::DirectoryContentsList
* juce::TimeSliceThread
* PlaylistComponent

**Methods:**

* FileBrowserComponent(PlaylistComponent& playlist)
* void resized()
* void selectionChanged()
* void fileClicked(const juce::File& file, const juce::MouseEvent& event)
* void fileDoubleClicked(const juce::File& file)
* void browserRootChanged(const juce::File& newRoot)
* void scanAndAddFiles(const juce::File& directory)

**OOP principles:**

* **Encapsulation:** the class encapsulates functionality for file system display and file selection handling.
* **Composition:** uses objects of PlaylistComponent, juce::FileTreeComponent, juce::DirectoryContentsList, and juce::TimeSliceThread.

**PlaylistComponent**

**Used classes:**

* juce::Component
* juce::TableListBoxModel
* juce::Button::Listener
* DeckGUI
* TrackInfo

**Methods:**

* PlaylistComponent(DeckGUI\* deckGUI1, DeckGUI\* deckGUI2)
* ~PlaylistComponent()
* Various overridden methods for table rendering and event handling.

**OOP principles:**

* **Inheritance:** inherits from juce::Component, juce::TableListBoxModel, and juce::Button::Listener.
* **Encapsulation:** manages playlist functionality, including track addition and button event handling.
* **Composition:** Uses DeckGUI and TrackInfo for interaction.

### Interaction between components

* FileBrowserComponent interacts with PlaylistComponent to add tracks.
* PlaylistComponent interacts with DeckGUI for playback control.

**InfiniteRotarySliderComponent implementation**

The InfiniteRotarySliderComponent is a custom JUCE component inspired by [this repository](https://github.com/deorst/InfiniteRotarySlider). The component allows infinite rotation, enabling track scrubbing.

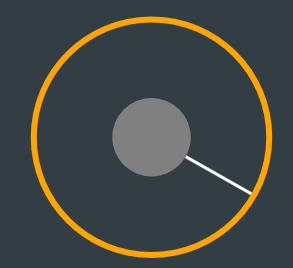


Image 6. View of InfiniteRotarySliderComponent in OtoDecks.

**Additional implemented methods:**

* void setAngle(float newAngle);
* void setOneRevolutionSliderMin(float newMin);
* void setOneRevolutionSliderMax(float newMax);
* void setEndAngle(float newEndAngle);
* void setStartValue(float newStartValue);
* void setStartValueNeedUpdate(bool newStartValueNeedUpdate);
* void setStep(float newStep);
* void setAdjustedValue(float newAdjustedValue);

### Timer-Based fnimation

The DeckGUI::timerCallback() method was modified to update the deck wheel position dynamically:

void DeckGUI::timerCallback()

{

double positionRelative = player->getPositionRelative();

waveformDisplay.setPositionRelative(positionRelative);

if (player->isPlaying())

{

double positionInSeconds = player->getPositionInSeconds();

infiniteRotarySlider.setValue(static\_cast<int>(positionInSeconds \* speed \*

180));

// Calculate the angle based on the relative position

float angle = static\_cast<float>(positionInSeconds \*

juce::MathConstants<double>::pi \* speed);

infiniteRotarySlider.setAngle(angle);

}

}

This implementation ensures real-time updates of the rotating deck wheel based on track playback position.

## Waveform display Click-to-Seek functionality

Implemented in WaveformDisplay:

* mouseDrag handles user interaction to update track position.
* onPositionChanged notifies DeckGUI, which updates player->setPositionRelative(newPosition) accordingly.

## Conclusion

This small DJ application provides essential functionality for track playback and management. Future work includes implementing a scratching effect for the InfiniteRotarySliderComponent, which currently allows track position adjustments but lacks full scratching support.

**References**

1. **JUCE Documentation** https://juce.com/learn/documentation/
2. **VirtualDJ UI/UX Research** YouTube video: " Virtual DJ Live Mashup" https://www.youtube.com/watch?v=GIPHwnokWiw&t=1149s
3. **GitHub Repositories for Slider Components** GitHub repository: "Infinite Rotary Slider by deorst" https://github.com/deorst/InfiniteRotarySlider
4. **Various DJ Software Interfaces** YouTube video: " FREE DJ Apps: Our Top 3 Picks for iPad and iOS" https://www.youtube.com/watch?v=gaVJ-8qjulI&t=1s