Project Documentation

Media Player

Overview

The goal of this project is to implement a media player application which can be used to play audio files of some format. The supported audio file formats include mp3, wav, and flac. The application is implemented as a Linux desktop application with the C++ programming language and some libraries, including Qt, the C++ standard template library and Taglib. With the application, user can open a directory of audio files, which are saved on the computer. The files are listed in the graphical user interface of the application. The list includes the metadata of the audio files, including track title, album and artist. The user can use the graphical user interface to control the playback of audio files. The playback can be controlled by pausing or resuming the currently playing track, or skipping to the next or previous track. Double-clicking a track in the list view skips immediately to the selected track. The application shows the track title, artist and the duration of the currently playing track. The interface has a slider which can seek to a specific point in the currently playing track, another slider for controlling the volume and a mute button.

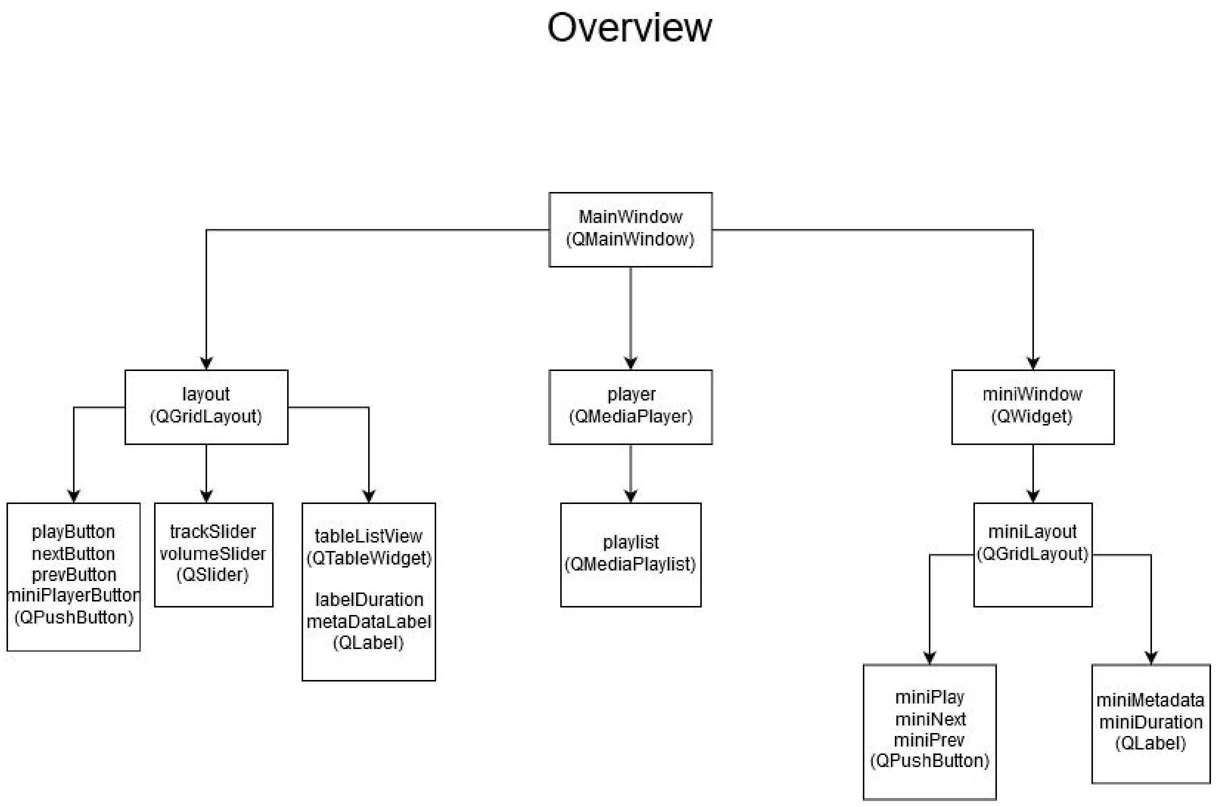
Scope

The media player application will have a minimal set of features which are required for playing audio files. This is not only due to the limited time and resources of this project but also a design objective of the application. The features are grouped into two categories: main features and additional features. The main features are considered crucial. They are needed in the application for it to be useful in any way. The additional features are a minimal set of features which make using the media player application significantly easier, but can be considered optional in case the project runs out of time or resources.

The main features of the application are the ability to load and play an audio file and a graphical user interface with playback controls. The playback controls include pausing, seeking and controlling volume with buttons and sliders. Mp3, wav and flac audio formats are supported by the application.

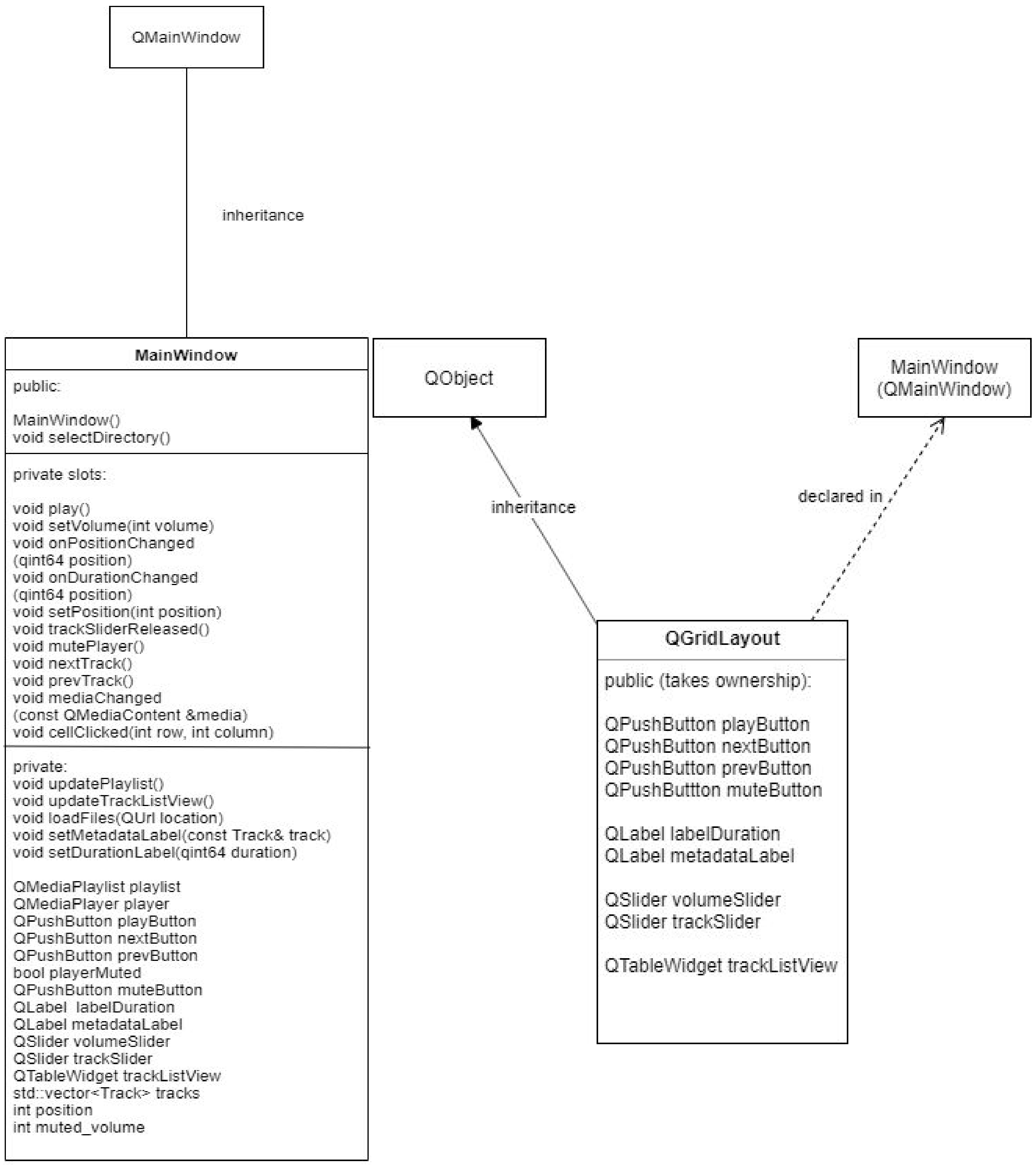
Some additional features were implemented during this project. In addition to the basic functionality, the application can display audio information. This information is read from the metadata of the audio file and it includes track, album and artist names. The application can browse and load audio files from a filesystem directory on the computer. The application loads files from the selected directory and its subdirectories recursively. Loaded tracks are listed within the application in a list. The list has columns for some track metadata attributes, such as track name, artist and album. The last implemented feature is a mini-player mode where the application hides most UI components and shows only playback controls and information about current track.

Software structure


Overview
MainWindow (QMainWindow)
layout (QGridLayout)
player (QMediaPlayer)
miniWindow (QWidget)
playButton nextButton prevButton miniPlayerButton (QPushButton)
trackSlider volumeSlider (QSlider)
tableListView (QTableWidget) labelDuration metaDataLabel (QLabel)
playlist (QMediaPlaylist)
miniLayout (QGridLayout)
miniPlay miniNext miniPrev (QPushButton)
miniMetadata miniDuration (QLabel)

Overview of the project's class hierarchy. It should be noted that in Qt, a class may *take ownership* of a previously declared class, meaning that the class which took ownership is responsible for memory management of its children. Also noteworthy is that in Qt, many data types such as QString are in fact classes.

The directory structure of the project is simple. The source code is in "src" directory and it is divided into four C++ files and three header files. Each header file contains one class definition or a function declaration. The project is built with CMake.



This project uses primarily two external libraries: Qt and Taglib. Qt is used as a framework to create the user interface of the application. The main interface between the user interface and other parts of the application is Qt signals which are used to call methods when the user interacts with the user interface. Functionality provided by the Qt framework is also used for other tasks such as finding audio files in a directory. Taglib is used for parsing file metadata from audio files.

Instructions for building and using

Required libraries and software

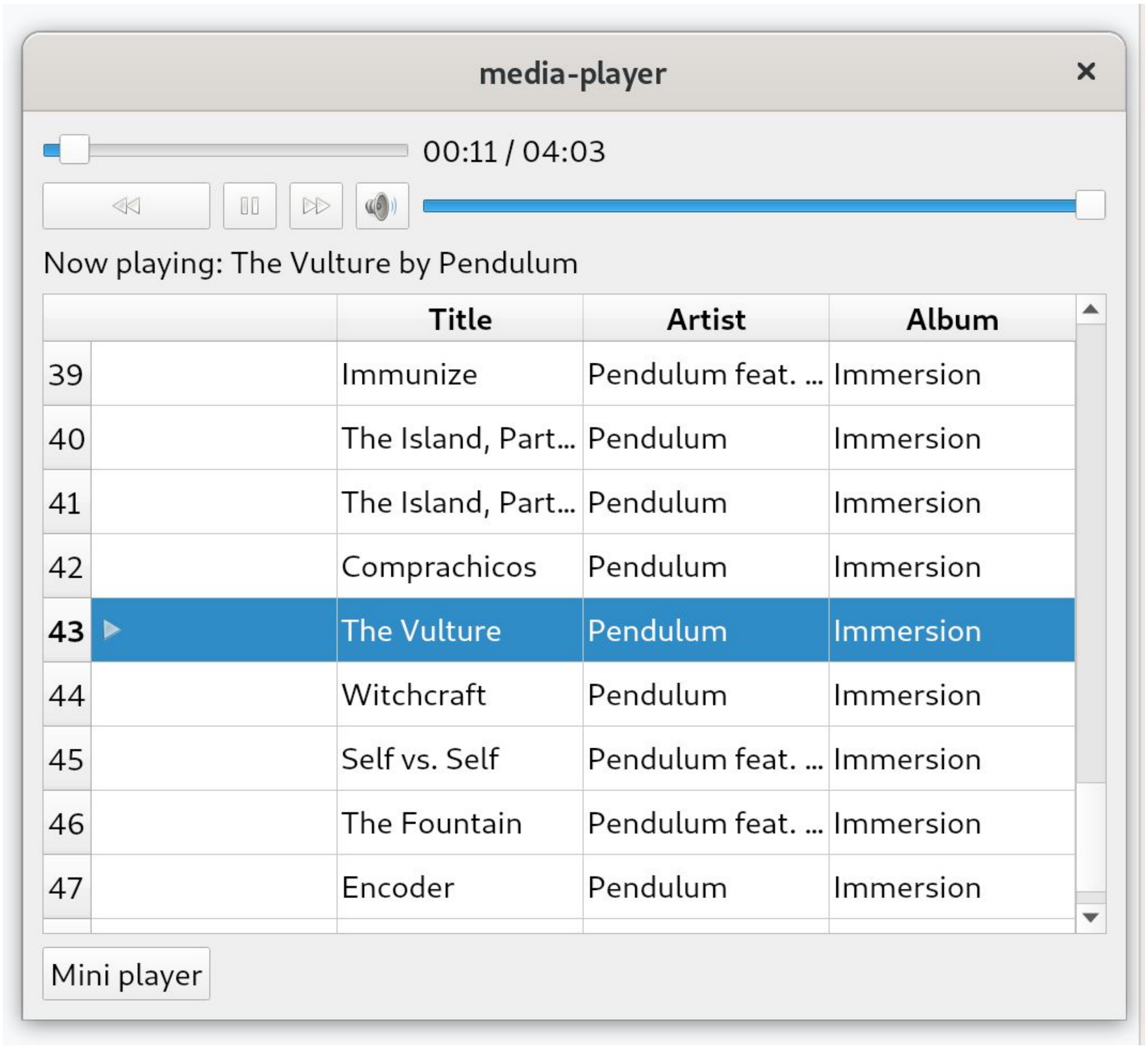
Building the application requires a C++ compiler, CMake and Make tools. The application uses Qt, Qt multimedia module and Taglib libraries. Taglib is downloaded and built by CMake. Taglib requires zlib. All dependencies can be installed on Ubuntu Linux by installing build-essential, cmake, qt5-default, qtmultimedia5-dev and zlib1g-dev packages. Some additional packages, such as GStreamer plugins, may be required to support all audio formats. Qt multimedia module is not available on Aalto Linux computers.

Building the application

First, make sure all development dependencies are installed and clone the project repository. Within the repository root, run ‘`cmake .`‘ command. Then run ‘`make`’ to build the application. This command takes slightly longer on the first run since it downloads and compiles taglib. The built application is saved as ‘`bin/media-player`‘ file in the build directory.

Using the application

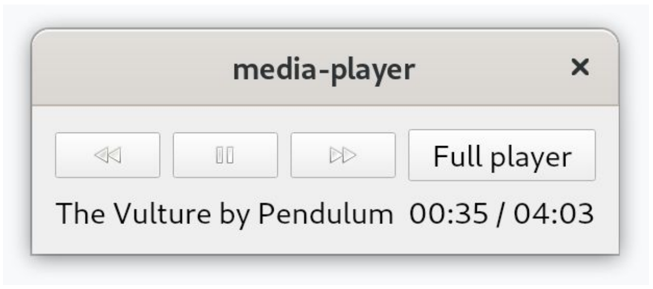
Start the application by executing ‘`./bin/media-player`’. The player will first prompt to open a directory which contains audio files. The files can be located in subdirectories of the selected directory. The application opens files with mp3, flac and wav file extensions.



Now playing: The Vulture by Pendulum

		Title	Artist	Album
39		Immunize	Pendulum feat. ...	Immersion
40		The Island, Part...	Pendulum	Immersion
41		The Island, Part...	Pendulum	Immersion
42		Comprachicos	Pendulum	Immersion
43	▶	The Vulture	Pendulum	Immersion
44		Witchcraft	Pendulum	Immersion
45		Self vs. Self	Pendulum feat. ...	Immersion
46		The Fountain	Pendulum feat. ...	Immersion
47		Encoder	Pendulum	Immersion

The full player mode has playback controls, including next, previous, play and pause buttons, and a track slider. The view shows track title, artist and duration of the currently playing track in the view and a list which includes title, artist and album of all loaded tracks. The track list highlights currently playing track. The track can be switched by double-clicking any track in the list. The view also has a slider for volume control and a mute button next to it. *Mini player* button opens a minimalistic version of the player and closes the full player view.



The mini player view has only next, previous, play and pause buttons for controlling playback. It also shows title, artist and duration of the current track. *Full player* button closes reopens the full player view and closes mini player.

Testing

Since most of the application code is user interface code, it has been tested by using the application. The testing procedure involves verifying that each feature of the user interface works as expected and the application does not crash. A simple testing procedure involves starting the application, opening a folder with some audio files and starting the audio playback. Then, mute, volume control, track slider, and playback controls can be tested. Finally, testing playback controls can be repeated in mini-player mode. The track list in full player mode should show correct metadata for each track and it can be verified during the playback that it matches to the track which is playing at the moment. During the testing, audio metadata should change when track changes, either when the current track ends, the track is changed from track list, or by clicking next or previous buttons.

The app has also been tested with invalid input, such as empty files with an audio file extension. In this case, the app shows an empty entry in the file list and it will not play the file. Some components log errors in the standard output but the app does not crash. Pressing next and previous in the ends of the whole playlist will only clear the metadata on the screen, and do not cause other visible effects on the application. Moving sliders while playing audio has been tested. Moving the track slider has been implemented so that it adjusts the track position only when the slider is released so it does not cause any audio effects. Closing the app in mini-player mode works as expected and the application quits.

Work log

Division of work and responsibilities

All team members participated in the implementation of the project. All team members tested the application and reviewed code written by other team members. The work and responsibilities were divided among team members by features of the application roughly as follows.

Atte Lautanala

- Project manager
- Setting up project codebase with CMake and Qt
- Audio playback and controls
- Browsing and loading audio files
- Listing opened audio files

Joona Kukkonen

- Displaying audio metadata
- Audio playback and controls
- Support for different audio formats
- Volume controls

Joonatan Syrjänen

- User interface layout
- Displaying audio metadata
- Browsing and loading audio files
- Volume controls

Väinö Kurula

- User interface layout
- Audio playback and controls
- Browsing and loading audio files
- Listing opened audio files
- Mini-player mode

Week 1 (26.10. - 1.11.)

During the first week, the initial project plan was created.

Hours used (approx.)

Atte Lautanala: 6 hours
Joona Kukkonen: 2 hours
Joonatan Syrjänen: 2 hours
Väinö Kurula: 3 hours

Week 2 (2.11. - 8.11.)

The second week was spent on creating a codebase for the project and each team member set up their development environments for the project. Audio playback was tested successfully with Qt Multimedia module.

Hours used (approx.)

Atte Lautanala: 3 hours
Joona Kukkonen: 6 hours
Joonatan Syrjänen: 10 hours
Väinö Kurula: 2 hours

Week 3 (9.11. - 15.11.)

Initial layout for the user interface was created and the first audio playback controls were added. The implementation of audio file browsing was started.

Hours used (approx.)

Atte Lautanala: 5 hours
Joona Kukkonen: 4 hours
Joonatan Syrjänen: 4 hours
Väinö Kurula: 3 hours

Week 4 (16.11. - 22.11.)

The most important features of the application, including audio playback, controlling the playback and browsing files, were finished in preparation for the mid-term meeting.

Hours used (approx.)

Atte Lautanala: 6 hours
Joona Kukkonen: 4 hours
Joonatan Syrjänen: 5 hours
Väinö Kurula: 5 hours

Week 5 (23.11. - 29.11.)

The work on displaying audio metadata was started. This included linking an additional audio file metadata library to the project. The first pieces of track information was added to the user interface, including information about current track and track duration. The work on loading audio files from directories and listing them in the user interface was started.

Hours used (approx.)

Atte Lautanala: 7 hours
Joona Kukkonen: 6 hours
Joonatan Syrjänen: 4 hours
Väinö Kurula: 4 hours

Week 6 (30.11. - 6.12.)

Displaying audio metadata in the user interface was finalised. The work on loading multiple audio files and listing all files with metadata was continued.

Hours used (approx.)

Atte Lautanala: 3 hours
Joona Kukkonen: 3 hours
Joonatan Syrjänen: 0 hours
Väinö Kurula: 4 hours

Week 7 (7.12. - 13.12.)

Loading audio all files from a directory was finished and listing the audio files in the user interface was implemented. Mini-player feature was added to the application. Some final tweaks were made in preparation for the project deadline, and the final project documentation was created.

Hours used (approx.)

Atte Lautanala: 9 hours
Joona Kukkonen: 1 hours
Joonatan Syrjänen: 2 hours
Väinö Kurula: 5 hours