Project Plan

Media Player

The goal of this project is to implement a media player application which can be used to play audio files of some format. Possible 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 and the C++ standard template library. With the application, user can open one or more audio files, or a directory of them, which are saved on the computer. The files are listed in the graphical user interface of the application. The list includes metadata of the audio files, such as track name, album and artist. The list can be sorted by some metadata attributes and the user can search for a specific track in the list. 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. The application shows metadata 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 and another slider for controlling the volume.

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, sliders and possibly other user interface controls. At this point, the application will support at least one audio file format. Additional file formats may be supported at early stages of the project if the library used for playback has good support for many file formats. Audio file formats which could be supported include mp3, wav and flac. However, this feature is considered optional, so it may be implemented with other additional features. These features are implemented before any additional features as they are the most crucial features of a media player application.

Some additional features will be 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 should include columns for some track metadata

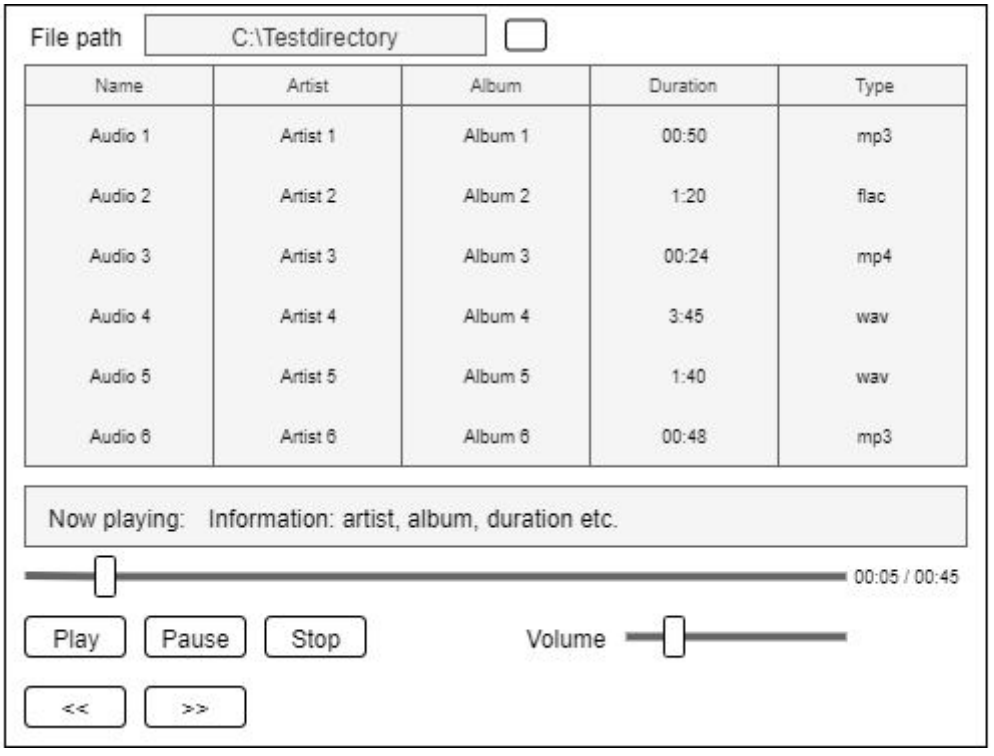
attributes, such as track name, artist and album. The list can be sorted by some metadata attributes and filtered. The filtering is based on a keyword given by the user, and the interface shows tracks which contain the keyword.

Preliminary schedule

The project starts with planning which will be completed by Friday 1st of November. The implementation of the application starts after this date. By the mid-term meeting, the goal is to set up the project, build system and libraries, and implement the basic features of the application. After the mid-term meeting, the focus is on additional features. Since there are some dependencies between these features, some of them must be implemented in specific order. More specifically, loading audio files from a folder should be implemented before implementing the track list. The list is also needed before track sorting or filtering. Displaying audio metadata and support for additional audio file formats can be implemented independently from other features after the basic features have been implemented.

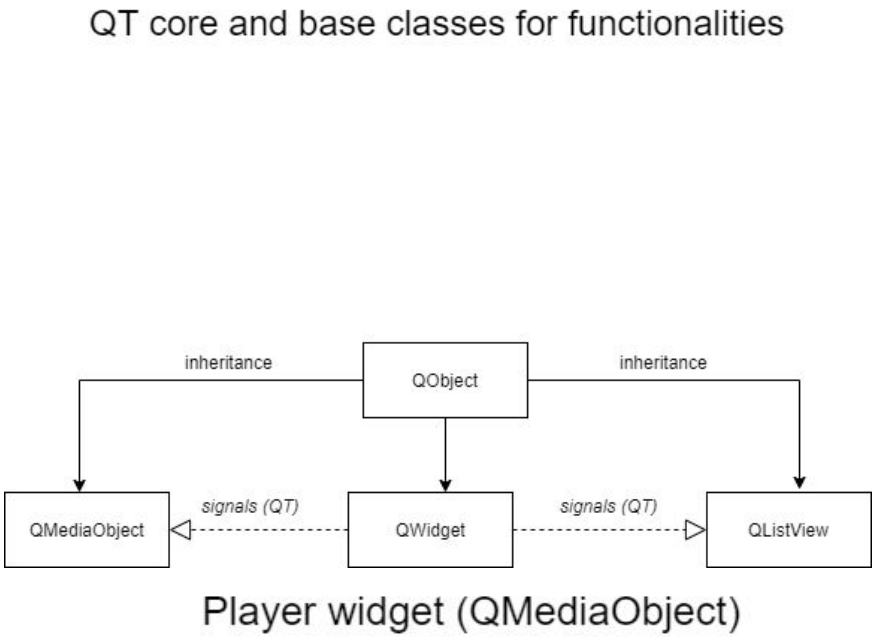
High-level structure

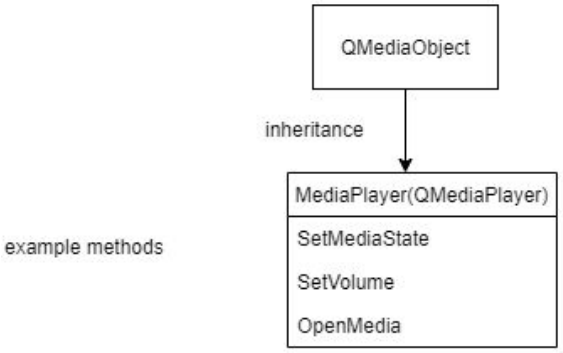
The user interface consists of one main view which includes all functionality planned for the application.



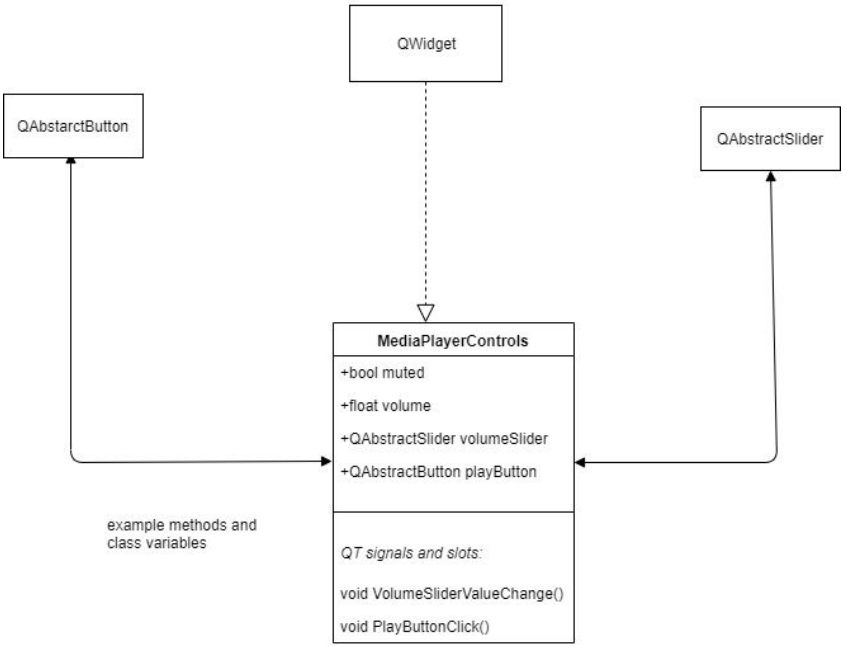
The diagrams below illustrate the functionality of the user interface as objects. It also includes the relationship of those objects to the Qt library which will be used in the implementation. Algorithms from the C++ standard template library will be used to implement some features, such as sorting and filtering audio files.

As shown in the below diagram, QT code contains a couple of keywords that are not standard to C++. The two most important of these (regarding this project) are *signals* and *slots*, which basically allow communication between objects. Signals and slots are a central QT alternative to callback techniques used in standard C++. Signals and slots allow other objects to be notified of a state change in the program. An example of where this is necessary is the *play*- button in the media player UI: the *play*- button, a separate object from the actual *MediaPlayer*, *emits* a *signal*, which is then received by the *MediaPlayer* object and causes the corresponding function within the player object to execute. A *slot* is the functionality of an object which reacts to emitted *signals.*

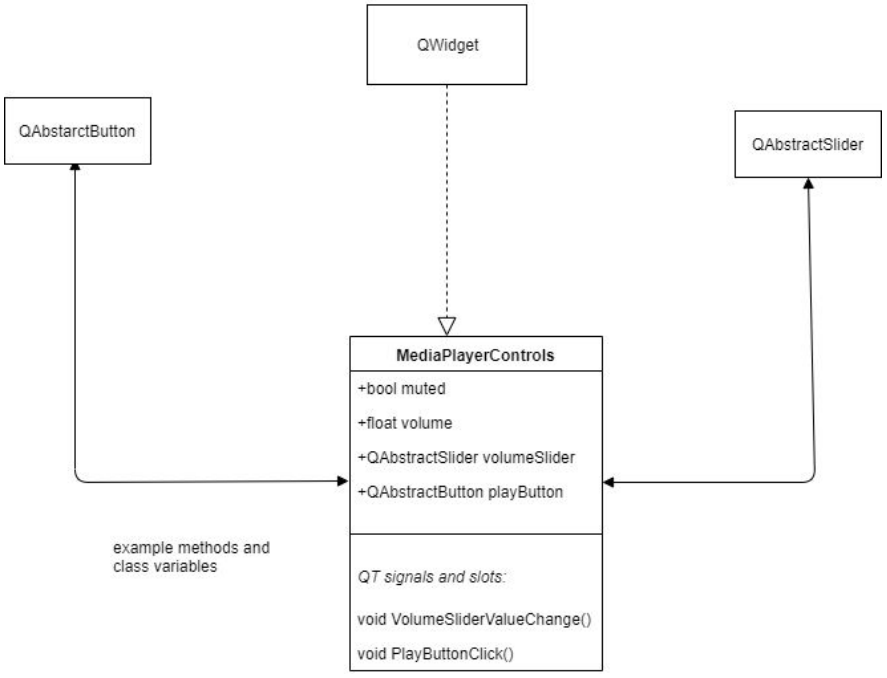




Media player buttons



Media player buttons



External libraries

In addition to the C++ standard template library, this project will use Qt to create graphical user interfaces. Since the Qt library is very extensive and contains facilities for more than only graphical user interfaces, it can be used to play audio files. Qt is also cross-platform by design and allows compiling for many platforms simultaneously. FFmpeg may be evaluated as an alternative library for audio playback. Catch2 or Googletest can be used for testing domain logic of some features, such as sorting and filtering.

Division of work and responsibilities

All team members will participate in the development of this project by implementing, testing and designing parts or complete features of the application. One of the team members, Atte L., is also the project manager. The additional responsibilities of the project manager include facilitating weekly meetings, tracking the status of the project and making sure that all team members know what to do. Meetings with the whole team will be organized weekly to facilitate division of work and address issues which may emerge during the implementation of the application. A Trello board will be used to track the status of tasks throughout the project.