OPP v0.5-dev

Maintenance Manual

Denis SAUNIER, Geoffrey BERGE,
Thibaud LAMARCHE, Thomas BERTHOME

June 04, 2014

Referring professor: Agnès ARNOULD

Diffusion list

Diffusion list		
Organism	**Name of the receiver**	**E-mail adress**
Cinema ouvert	Robin KRIER	robin@cinemaouvert.fr

Version follow

Document Follow			
Version	**Date**	**Name of the author**	**Comments**
1.0	06/06/14	Thibaud Lamarche	Initial version of the document
1.0	09/06/14	Thomas Berthomé	Added core class description
1.0	09/06/14	Denis Saunier	Added maintenance and deployment
1.0	09/06/14	Geoffrey Bergé	Added User Interface class description

Summary

Introduction

Project background

We were asked to take over the realization of a movie management software for the non profit organization "Catalogue Ouvert du Cinéma" (COC). The main goal of this organization is to promote independent cinema such as : movies under free license, movies moved to public domain or even short films. This project is based on the fact that, cinema associations are often making projections without necessary tools. They usually end up using a simple personal laptop with VLC on it to project the films, which usually creates some problems such as : the projectionist desktop showing on the screen or difficulties to set the good audio balance between movies.

The main goal of this project is then to carry on with the software development, to correct, and finalize an existing software in order to allow a projectionist to manage a play-list of movies which will be displayed on a screen. The projectionist would have a little sample of the running movies on his screen in order to have a feedback of the projection. He should be able to manage the sequence of the projection, the programming and the specific audio balance for each movie.

Constraints

This project contains some constraints :

- in order to evolve (with the community help), the software must be released under a free license (GNU GPL License).
- the software must deal with multiple platforms such as Linux, Mac OS and Windows, in this specific order because Linux is aimed at the programmers community to allow further development, Mac OS because it is used by a large portion of projectionists and finally Windows to ensure that the software is available for a larger group of people.
- The taking over of the existing project started by last year's group with all the difficulties thus generated.

Analysis

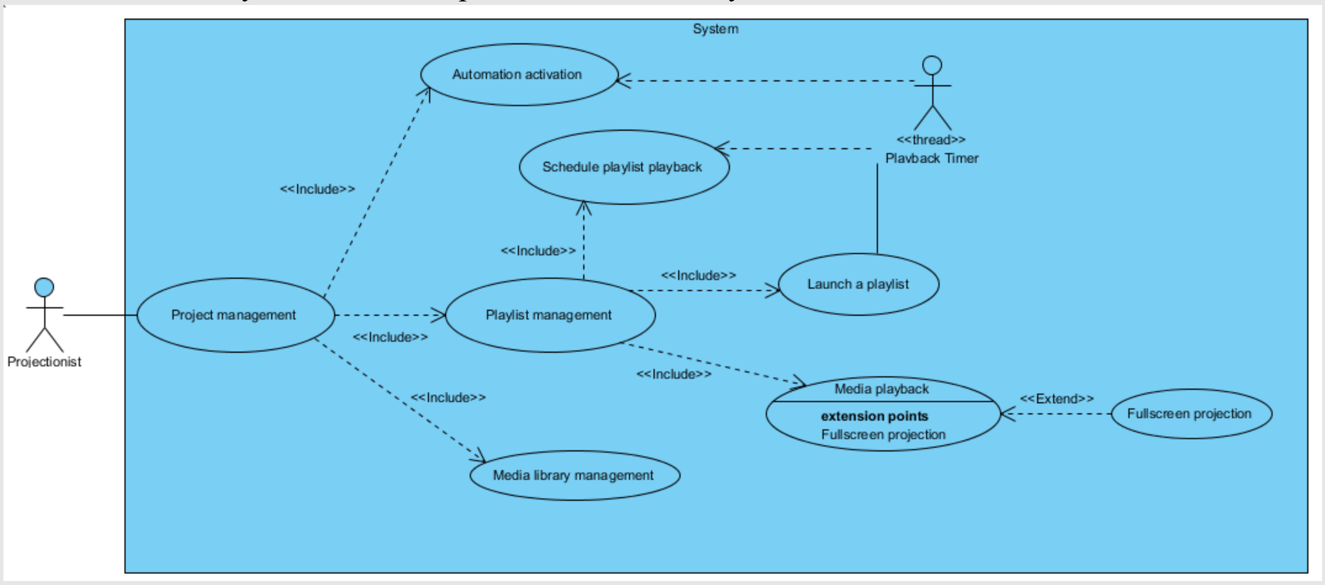
A year ago a group of students of the university of Poitiers started this project. This is their analysis of the project.

First analysis

Quote:

Authors : Baptiste Roziere, Cyril Naud, Florian Mhun, Hamza Haddar

The figure below represents our use case diagram, which illustrates the different use cases and main functions of our system. Also, it represents the main ways to use the software.

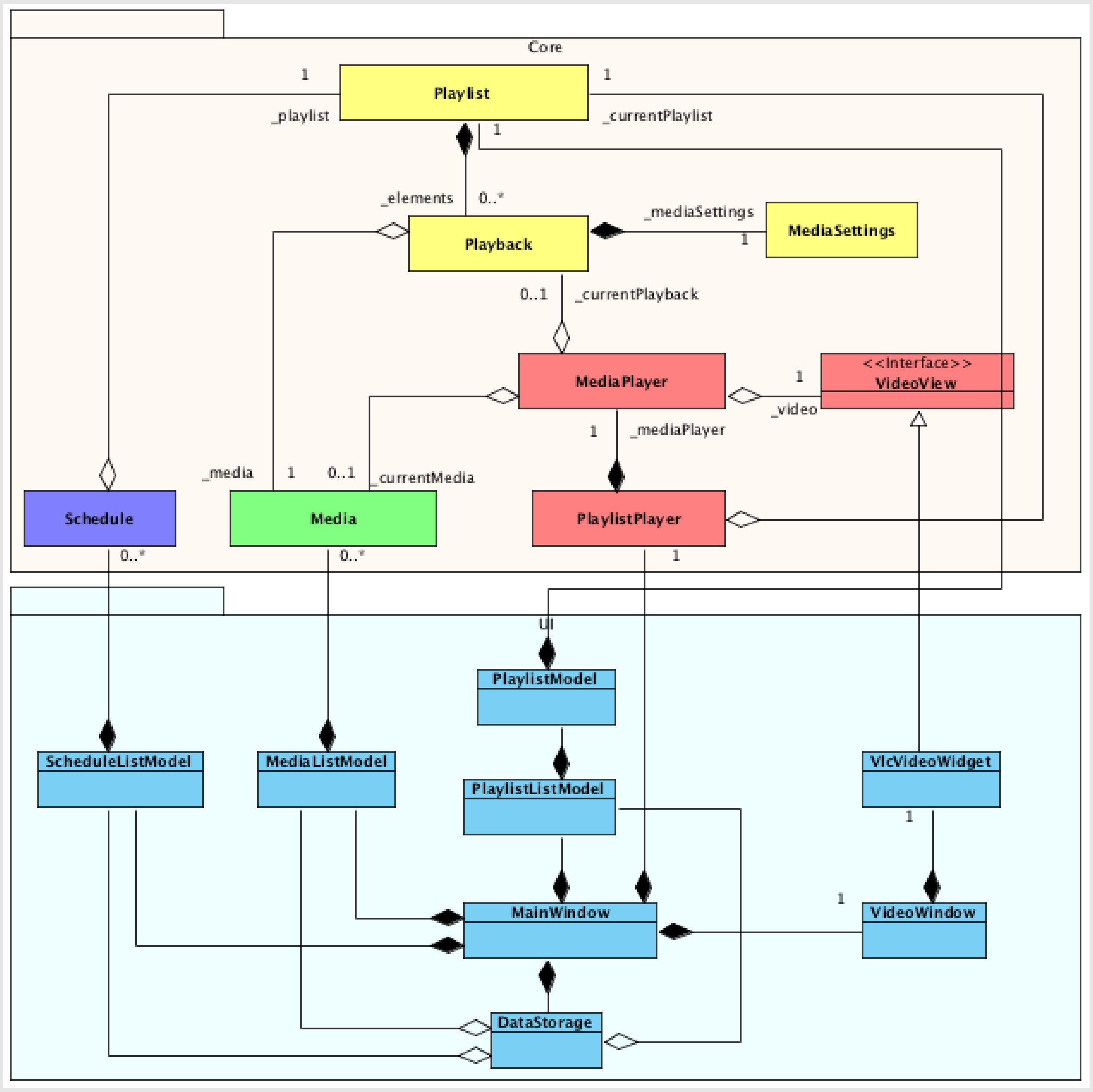


Project management, playlist management, schedule playlist playback, automation... represent the different use cases (functions) of our system. We can see that there are relationships between them. For example, the relationship “include” between Playlist management and Media playback means that, to achieve the objective Playlist management, we use the objective (use case) Media playback. And as we can see, we have chosen to represent the timer that triggers the playing of playlist as an actor.

Class diagram

Quote:

Authors : Baptiste Roziere, Cyril Naud, Florian Mhun, Hamza Haddar

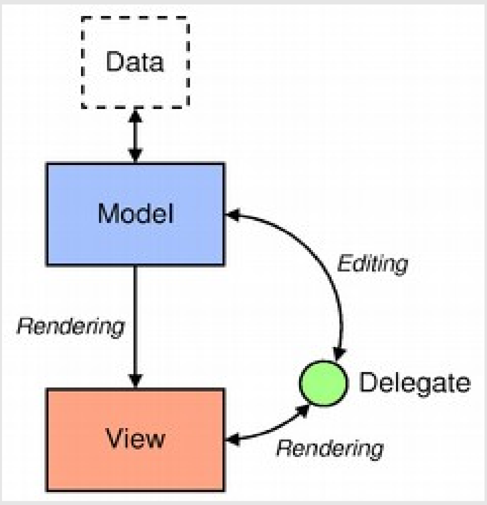


Model View using Qt

Quote:

Authors : Baptiste Roziere, Cyril Naud, Florian Mhun, Hamza Haddar

The Qt model view is designed as follow :



This concept is mainly used to render lists with Qt. There are three entities, the model, the view and, optionally, a delegate. Technically, we implement the model/view by subclassing abstract Qt classes and overloading methods.

The model is bound to a data source. In our case, data were media list, schedule list, playlist (list of playback). We have not yet implemented the model/view for playlist list but we are working on it.

The model provides methods that change data state like insert, remove, modify and sort items.

When the data changes, the model automatically applies the rendering into the view.

The view is just responsible of rendering data into a Qt component. We used only QTableView to display data in a table but other components exists, like QTreeView which can render data with a hierarchical structure.

The delegate is not required to deal with model/view programming. In fact, sometimes you need to handle behavior between the model's data state and the view rendering. The delegate acts as a "proxy" object between the model and the view.

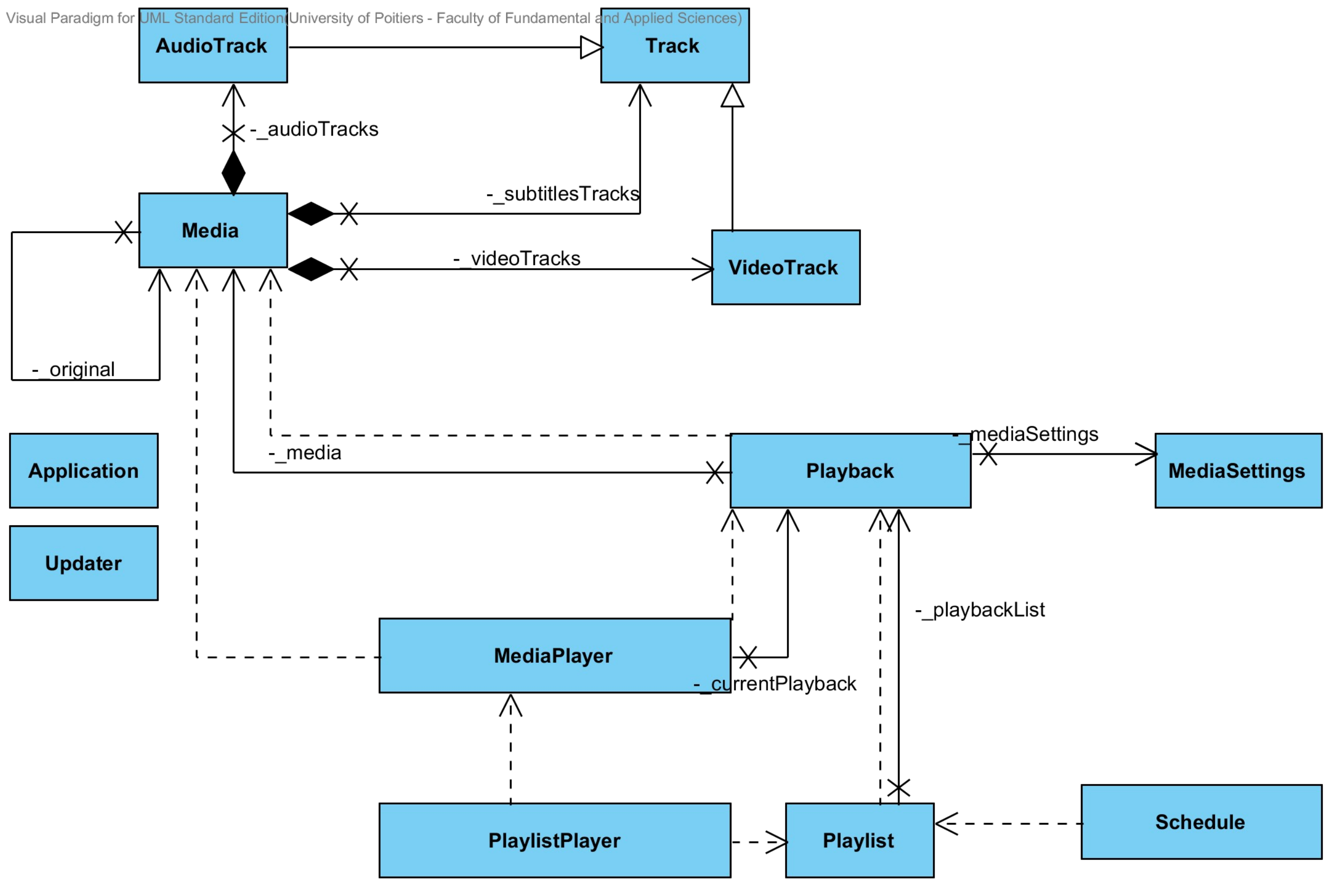
For example, when the user double clicks onto an item, the delegate can turn the item label text into a combobox and let the user edit the entry. Then when the user submits a change, the delegate is responsible to modify item's data through the model.

Current analysis

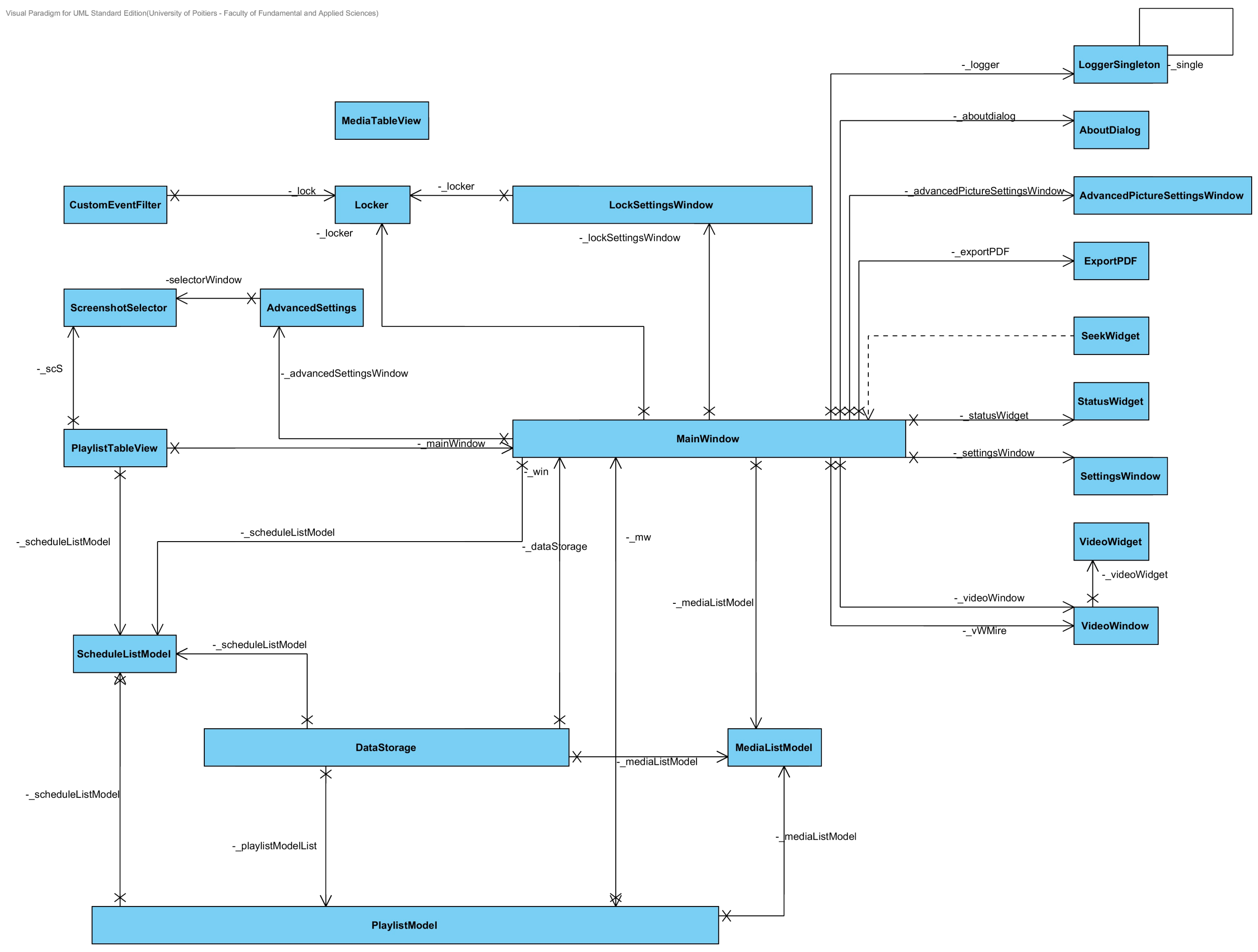
As you can see in the first analysis the software is separated in two parts, the core and the user interface.

Class diagram

Core



User interface



Class description

Core

Application

Name	application.h / application.cpp
Description	The class Application is used to manage global application settings. It created the instance of VLC with some arguments.
Interaction	Used by : DataStorage,ExportPDF, Main, MainWindow, MediaPlayer, MediaTableView, PlaylistTableView, SettingsWindow, VideoWindow
Critical aspect	

Audiotrack

Name	audiotrack.h / audiotrack.cpp
Description	The class AudioTrack inherits of Track. It is used to manage audio track informations. It differs a VideoTrack than an AudioTrack.
Interaction	Used by : AdvancedSettingsWindow, DataStorage, MainWindow, Media, MediaPlayer, MediaSettings, Playback and PlaylistModel
Critical aspect	

Config

Name	config.h
Description	The class Config is a static class. It defines some configuration like the version number, the link for update or the name of the installer...
Interaction	Used by : AboutDialog, Main and Updater
Critical aspect	Take care of multiplatform implementation.

Media

Name	media.h / media.cpp
Description	The class Media is used to manage media informations.. It defines the image extensions, the audio extensions and the video extensions. It prepare the media to be played by the vlc instance.
Interaction	Used by : AdvancedSettingsWindow, DataStorage, MainWindow, MediaListModel, MediaPlayer, Playback, Playlist, PlaylistModel, PlaylistTableView, ScreenshotSelector
Critical aspect	

MediaPlayer

Name	mediaplayer.h / mediaplayer.cpp
Description	The class MediaPlayer is used to manage media playback. It applies media settings on the media. It manage the back window (streaming and screenshots) and the fades.
Interaction	Used by : MainWindow, PlaylistyPlayer, SeekWidget
Critical aspect	5 timers are implemented (4 for fades, one for screenshots).

MediaSettings

Name	mediasettings.h / mediasettings.cpp
Description	The class MediaSettings is used to manage media settings. It defines some enumerations like ratio, desinterlacing or scale. It contains all settings vlc.
Interaction	Used by : AdvancedPictureSettingsWindow, AdvancedSettingsWindow, DataStorage, MainWindow, MediaPlayer, Playback, PlaylistModel.
Critical aspect	MediaSettings emit signals at every change.

Playback

Name	playback.h / playback.cpp
Description	The class Playback is used to manage playback by associating a MediaSettings and a Media instance.
Interaction	Used by : AdvancedPictureSettingsWindow, AdvancedSettingsWindow, DataStorage, MainWindow, MediaPlayer, Playlist, PlaylistModel.
Critical aspect	

Playlist

Name	playlist.h / playlist.cpp
Description	The class Playlist is used to manage list of Playback.
Interaction	Used by : DataStorage, MainWindow, PlaylistModel. PlaylistPlayer, PlaylistTableView, Schedule, ScheduleListModel
Critical aspect	

Plugins

Name	plugins.h
Description	This file is the place where the plugin interfaces are declared.
Interaction	Used by : MainWindow
Critical aspect	

PlaylistPlayer

Name	playlistplayer.h / playlistplayer.cpp
Description	The class PlaylistPlayer is used to manage playback of Playlist. It is the link between the user actions on the ui and the MediaPlayer.
Interaction	Used by : MainWindow, MediaPlayer, PlaylistModel, PlaylistTableView.
Critical aspect	It use its parent, don't forget to cast the parent before use it.

Schedule

Name	schedule.h / schedule.cpp
Description	The class Schedule is used to manage Playlist playback scheduling.
Interaction	Used by : DataStorage, MediaPlayer, PlaylistModel, PlaylistTableView.
Critical aspect	It use a timer to connect the current time with the time the user wants launch the playlist.

Track

Name	track.h / track.cpp
Description	It is used as base class for track informations. It is the super class of AudioTrack and VideoTrack. It redifine operators "=" and "==".
Interaction	Used by : AdvancedSettingsWindow, AudioTrack, Media, MediaPlayer, VideoTrack
Critical aspect	

Updater

Name	updater.h / updater.cpp
Description	It is used as base class for update application. It check the current version of the software and if a newest is available. It also open the url where the newest version is.
Interaction	Used by : MainWindow
Critical aspect	

Videotrack

Name	videotrack.h / videotrack.cpp
Description	The class VideoTrack inherits of Track. It is used to manage video track informations. It created the instance of VLC with some arguments.
Interaction	Used by : AdvancedSettingsWindow, DataStorage, MainWindow, Media, MediaPlayer, MediaSettings, Playback and PlaylistModel
Critical aspect	

User Interface

AboutDialog

Name	aboutdialog.h / aboutdialog.cpp
Description	Manages the Qdialog named "About" accessible from the "Help" menu at the top of the interface.
Interaction	Used by the class MainWindow to initialize it and open the window when the user whishes.
Critical aspect	The number version used is declared in the config.h file.

AdvancedPictureSettingsWindow

Name	advancedpicturesettingswindow.h / advancedpicturesettingswindow.cpp
Description	Manages the Qdialog named "Advanced picture settings" accessible from the button "Advanced picture settings", the deinterlacing mode and the crop. Applies the settings selected by the user on the playback.
Interaction	Uses the class Playback for apply the settings. Used by MainWindow to initialize it and open the window when the user presses the button.
Critical aspect	

AdvancedSettingswindow

Name	advancedsettingswindow.h / advancedsettingswindow.cpp
Description	Manages the Qdialog named "Advanced settings" accessible from the button "Advanced settings", in/out mark, image duration and fade in/out. Applies the settings selected by the user on the playback.
Interaction	Uses the class Playback for apply the settings Uses ScreenshotSelector for open a screenshot selector if the user presses the button. Uses the convertion methods in the class Utils. Used by MainWindow to initialize it and open the window when the user presses the button.
Critical aspect	

CustomEventFilter

Name	customeventfilter.h / customeventfilter.cpp
Description	Manages the auto-locking with a QTimer.
Interaction	Uses the class Playback for apply the settings. Used by main.cpp.
Critical aspect	In the main, this class is used in the method installEventFilter() thus the customeventfilter receive all events. So just compare the type of event to do what you need.

Datastorage

Name	datastorage.h / datastorage.cpp
Description	Manages the save and the load of a listing.
Interaction	It uses classes that are related to the media, playback, the playlist and the automation. Used by the class MainWindow to initialize it, to save or load a listing when the user whishes.
Critical aspect	

ExportPDF

Name	exportpdf.h / exportpdf.cpp
Description	Manages the export of the automation in pdf and the QDialog named "Export PDF".
Interaction	Used by the class MainWindow to initialize it and open the window when the user whishes.
Critical aspect	The text is declared in HTML and exported in PDF.

Locker

Name	locker.h / locker.cpp
Description	Manages the lock of the playlists.
Interaction	Used by the class MainWindow and CustomEventFilter to lock the playlists.
Critical aspect	

LockSettingsWindow

Name	locksettingswindow.h / locksettingswindow.cpp
Description	Manages the QDialog named "Lock settings", the password for the lock and the duration for the auto-locking.
Interaction	Used by the class MainWindow to initialize it and open the window when the user whishes.
Critical aspect	

LoggerSingleton

Name	loggersingleton.h / loggersingleton.cpp
Description	Manages the QLabel used in the Log tab. The instance is unique (singleton).
Interaction	Used by the class MainWindow to initialize it and write the different errors in the QLabel.
Critical aspect	It's a singleton so there is a mutex in the methods because the instance is unique.

Main

Name	main.cpp
Description	Initializes the application, loads the settings, loads the translation, initializes the MainWindow, installs the custom event filter and shows the MainWindow.
Interaction	Uses the class MainWindow for initialize it and show it.
Critical aspect	

MainWindow

Name	mainwindow.h / mainwindow.cpp
Description	Initializes the different components of the main window and manages all the interactions between this window and the user.
Interaction	
Critical aspect	A method is defined for each possible user action.

MediaListModel

Name	medialistmodel.h / medialistmodel.cpp
Description	This class manages the model of the media tab. It creates the columns and manage the medium.
Interaction	Created in the mainwindow and added to the ui.
Critical aspect	

MediaTableView

Name	mediatableview.h / mediatableview.cpp
Description	Manages the mouse events of the media bin table
Interaction	Created in the mainwindow and gave to the mediaTableView
Critical aspect	

PlaylistModel

Name	playlistmodel.h / playlistmodel.cpp
Description	This class manages the model of the playlist tab. It creates the columns and manage the playbacks.
Interaction	Created in the mainwindow and gave to the playlistTableView
Critical aspect	

PlaylistTableView

Name	playlisttableview.h / playlisttableview.cpp
Description	Manages the mouse events of the playlist table
Interaction	Created in the mainwindow and gave to the playlistTableWidget
Critical aspect	

ScheduleListModel

Name	schedulelistmodel.h / schedulelistmodel.cpp
Description	This class manages the model of the schedule tab. It creates the columns and manage the schedules.
Interaction	It is created inside the mainWindow and passed to the scheduleTableView
Critical aspect	

ScreenshotSelector

Name	screenshotselector.h / screenshotselector.cpp
Description	This class launches a media inside a videoWidget in a custom Qdialog, it allows the user to navigate inside the movie to choose a screenshot to represent the media.
Interaction	Launched by the mainWindow when you do a right click on a media or launched from the advancedSettingsWindow when you want to change the screenshot.
Critical aspect	In order to navigate inside the media you need to start by playing it, that why the video mooves a litle at the launch.

SeekWidget

Name	seekwidget.h / seekwidget.cpp
Description	This class is used to display and manage a small seek bar (display the current time of the projected media, and allow the user to navigate inside it).
Interaction	Used by the mainwindow
Critical aspect	catch time events of libvlc in order to set the current time

SettingsWindow

Name	settingswindow.h / settingswindow.cpp
Description	This class is used to display a Qdialog containing the current settings of the software, and allowing the user to manage these.
Interaction	It's launched by the mainWindow, and modifies the Qsettings « opp »
Critical aspect	Store all settings informations inside the Qsettings « opp »

StatusWidget

Name	statuswidget.h / statuswidget.cpp
Description	This class is used to display a status bar at the bottom of the software.
Interaction	The statusWidget is owned by the mainWindow.
Critical aspect	A Qtimer (used to display the current hour) is launch when the constructor is call, and is always running after that.

Utils

Name	utils.h / utils.cpp
Description	This class is not a class (-_-'). It is used to deal with frequently needed methods like the transformation of msecs to Qtime
Interaction	
Critical aspect	

VideoWidget

Name	videowidget.h / videowidget.cpp
Description	This class is used to redirect the video stream from vlc default player to our custom widget
Interaction	It is created and hold by videowindow and passed to the PlaylistPlayer to redirect the stream.
Critical aspect	There is no multiplatform method to deal with this, so there is a block of code for each platform

VideoWindow

Name	videowindow.h / videowindow.cpp
Description	This class is used to create a window for the projection. It can be displayed in Window(small size) mode or in Projection mode(full screen).
Interaction	It is used three times in MainWindow. Once to create the video window for the projection another time to create the screenshots of the medium and finally to project the test pattern.
Critical aspect	Launch a closed signal when the user request the close of the window in order to properly stop the playback

Maintenance

Dependencies

This software uses the vlc library, it was tested with several libvlc versions (from 2.0.8 to 2.1.4). Then, it uses QtNetwork, to upgrade OPP, QtWebkit, to export the schedule to PDF, and QtXml, to save.

Compilation

The compilation can be done on Windows, Linux and MacOs with QtCreator.

For more explanation, take a look at this link:

How to compile

Plug-in

To create a plug-in, you must use the OPP library. Then, you must create a new project with a class that implements a plug-in interface declared in OPP.

For more explanation, take a look at this link:

How to create a plug-in

Change the version number

The version number of the software program is shown in the config.h file. To modify the version, you must change the value of the variable “VERSION”.

Update

To upgrade OPP, the software checks the version number from the release on the http://cinemaouvert.fr/ website.

The web tree is http://cinemaouvert.fr/update/ “NAME OF OS” / latest /

In the folder “latest”, you can find the installer and a file “version.txt” with the version number inside.

If you want to update the name of the installers, take a look at the file config.h.

Add a media setting (clamp to the media and saved)

To add a media setting like “gamma”, you must create a getter and a setter for the setting in MediaSetting class. Then, a signal for the setting like “`gammaChanged(gamma)`”.

In the MediaPlayer class, you must create a setter:

```
setCurrentGamma(float)
```

In this setter, you must call a libvlc method (the one you want to apply).

```
libvlc_video_set_adjust_float(_vlcMediaPlayer, libvlc_adjust_Gamma, gamma)
```

Then, you must connect the setting in the “`open()`” method:

```
connect(_currentPlayback->mediaSettings(), SIGNAL(gammaChanged(float)),
      this, SLOT(setCurrentGamma(float)));
```

In the “`close()`” method, you must disconnect the setting:

```
disconnect(playback->mediaSettings(), SIGNAL(gammaChanged(float)), this,
      SLOT(setCurrentGamma(float)));
```

Finally, you must add the setter in the “`applyCurrentPlaybackSettings()`” method.

To save the setting, you must add this line in the Datastorage class in a save method:

```
playback.setAttribute("gamma", playbackElement->mediaSettings()->gamma());
```

And to load the setting, you must add it in the “`load()`” method:

```
      settings->setGamma(
playbackAttributes.namedItem("gamma").nodeValue().toFloat());
```

License

The software is delivered under the version 3 of the GNU GPL license. For all details about GPL license, refer to the following page http://www.gnu.org/licenses/gpl.html

Deployment

Translation

The software use a translation system. The default language is English, the other languages can be integrated from translation files.

How to translate

Installers

We have created an installer for several operating system. For Windows, we generate an installer with “Inno Setup Compiler”, for Ubuntu/Debian a .deb with “Debreate” and for Mac OS a .pkg with “packageManager”.
How to create an installer

Generate the Doxygen

To generate the Doxygen documentation, you can download Doxywizard and install Doxywizard with Graphviz.

Download link: http://www.stack.nl/~dimitri/doxygen/manual/doxywizard_usage.html

Then, you must get doxyFile on your local repository of OPP and run Doxywizard with the doxyFile.

Users documentation

In order to properly launch the users documentation you need to place it into the folder « help » and name it « usersDocumentation.pdf ».