Cross Platform Media Center

(Audio Module)

Final Year project report submitted in the partial fulfillment of the requirement for the degree of

Bachelor of Technology

in

Computer Science and Engineering

under the supervision of

Mr. Suman Shah

By

Vardhan Jain (101208)

To

Jaypee University of Information and Technology

Waknaghat, Solan – 173234, H.P

Table Of Contents

CERTIFICATE

This is to certify that the work titled "**Cross Platform Media Center**" submitted by "**Vardhan Jain**" in partial fulfillment for the award of degree of B.Tech of Jaypee University of Information Technology, Waknaghat has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

Signature of Supervisor ……..

Name of Supervisor……..

Designation ……..

Date ……..

Acknowledgement

I would like to express our greatest gratitude to the people who have helped & supported us throughout our project. I would like to owe our heartiest thanks to our mentor **Mr. Suman Shah** for his continuous support for the project and providing guidelines for making this project.

Signature of student……..

Abstract

Cross-Platform Media Center is a Desktop Application which will contain five modules - Music, Video, Image, LAN Messaging and LAN Video Chat.Users can listen to music, watch videos, view images, can apply real-time video and audio effects and can also apply effects in images and can save them. Apart from these, one of the most exciting feature is LAN Video Chat. Users can Video Chat with one another via LAN connectivity.

I am designing the Audio module of this media center.

Chapter – 1

Introduction

Cross Platform Media Center

1.1)Introduction

Now a days, the main question that concerns majority of the people is their modes of entertainment after their long hours of hectic careers. They mostly engage themselves in listening to music, watching videos or movies, surfing websites and connecting themselves socially. Seeing the strong desires of our upcoming generations for recreation, many softwares developing companies focused on developing softwares and games that could fulfill the desires of our generations. Progressing further in the development, the idea to aggregate various entertaining stuffs in a single interface evolved. Taking this into consideration the idea of Media Center came up which involved a dedicated media player for various recreational stuffs like listening to music, watching videos or movies, viewing images etc. in a single interface. People had an advantage on these media centers as they had to get acquainted with just a single interface. Also the main motive of media centers was to look more interactive and user friendly so that people could feel the ease of using it.

As more and more development is required in every field, developers started to take views of the general people about what more new features they would want to add to the media centers to make it more recreational. So, the features like Online TV Shows, Chatting, Network streaming etc. were implemented and more media centers with added features were developed and are still being developed. People started liking them as they demand more and more stuffs in a single interface which saves their time and make them easily get acquainted with the interface.

Taking these points into focus, a lot of media centers have been made till date but a lot of them either misses some of the functionality or they are less interactive or have a lot of bugs. Missing functionalities may include lack of supported extensions, lack of some module or less customization.Many media centers are also platform dependent. So, there is still a lot of development required in this field.

1.2) Problem Statement

Many media centers which lack a lot of features in them are lack of extension support and platform independence. Different people use different platforms and so they want media centers of their choice to be platform independent. Making our media center cross platform will make it to run in all the major platforms. Also, I am trying to support most of the extensions in our media center whether it is for music, video or image. I am also adding the modules like LAN Messaging and LAN Video Chat which will make it more entertaining. There is no such media center upto date with such support.

Also, most of the media centers lack audio presets and live video effects.The addition of these functionalities in our media center will make it more entertaining and attractive. People will feel the worth of using such an application.

1.3)Objectives

The main objectives of my project is :

- To make an Audio Player.
  Player should support all the major audio extensions and functionalities such as playlists, presets, library management etc.

- To make a image viewer.

1.4)Approach

Iam making this project by using Qt Framework. Qt is based on C++ and has a rich APIs for GUIs, Networking, Graphics etc. that is cross-platform. The framework supports all the major platforms which may include Windows, Linux, Mac OS, Symbian, Solaris etc. The development is still going on in Qt and two stable versions are released every year with added features. It can also be ported to Android and iOS but this requires the installation of Qt libraries in the respective platform. The development is still going on to make it portable in these two operating systems natively and the stable version is expected to be released by September,2014.

The source code needs to be recompiled at different platforms to make the target Executable. With the rich support of GUI and Graphics libraries, our audio, video and image modules will be made by making use of these libraries. LAN Chatting and Video Conferencing will require the use of network APIs as well.

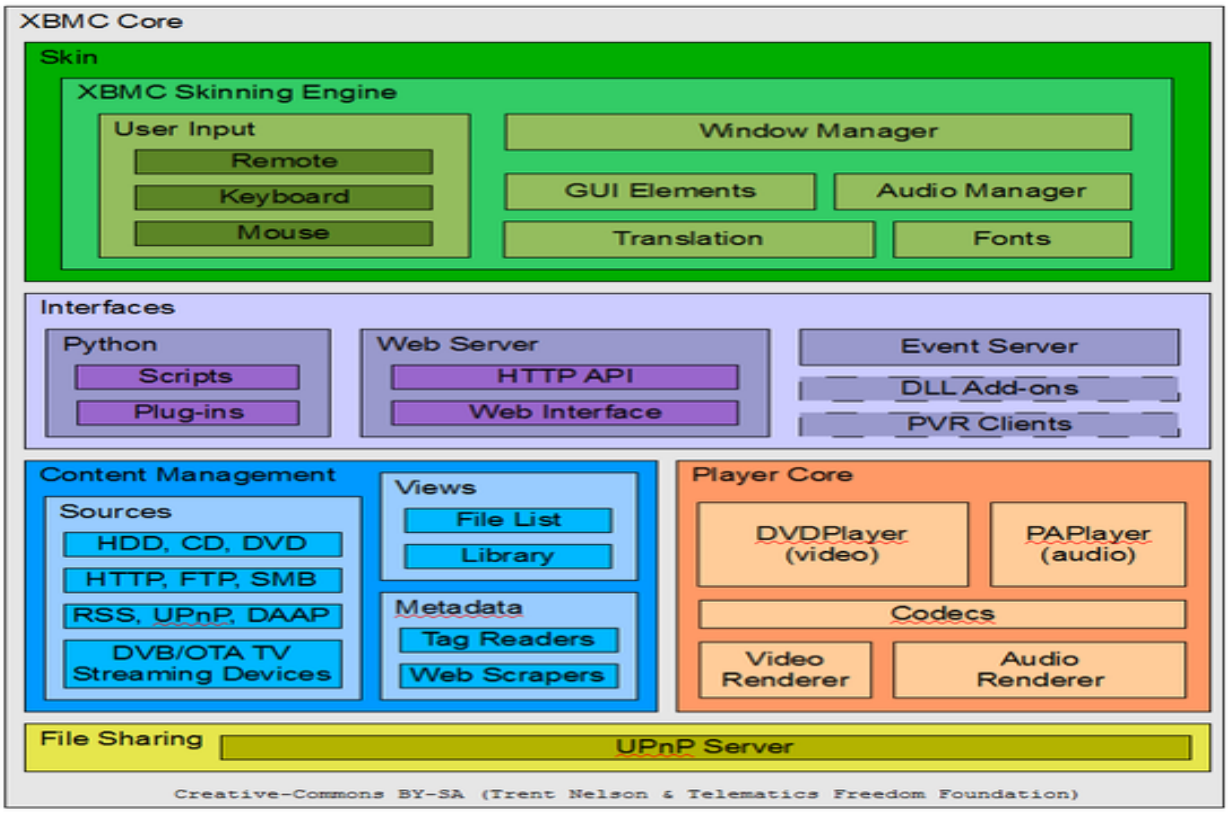
Chapter – 2

Literature Survey

Cross Platform Media Center

Comparisons

2.1.1). XBMC Media Center :



XMBC Media Center Architecture

XBMC is a free and open source media player developed by the **XBMC Foundation**, a non-profit technology consortium.XBMC is available for multiple operating-systems and hardware platforms, with a user interface for use with televisions and remote controls. It allows users to play and view most videos, music, such as podcasts from

the internet, and all common digital media files from local and network storage media.

The software was originally created as a media center application named "*Xbox Media Center*" for the original Xbox game console, but is today officially available, under the name "*XBMC*", as a native application for Android, Linux, BSD, Mac OS X, iOS, and Microsoft Windows operating systems.

Features :

1). Audio, video and pictures playback and handling.

2). Live TV with EPG and PVR/DVR Frontend.

3). Add-ons Manager.

4). Supports a large number of platforms and architectures.

Limitations :

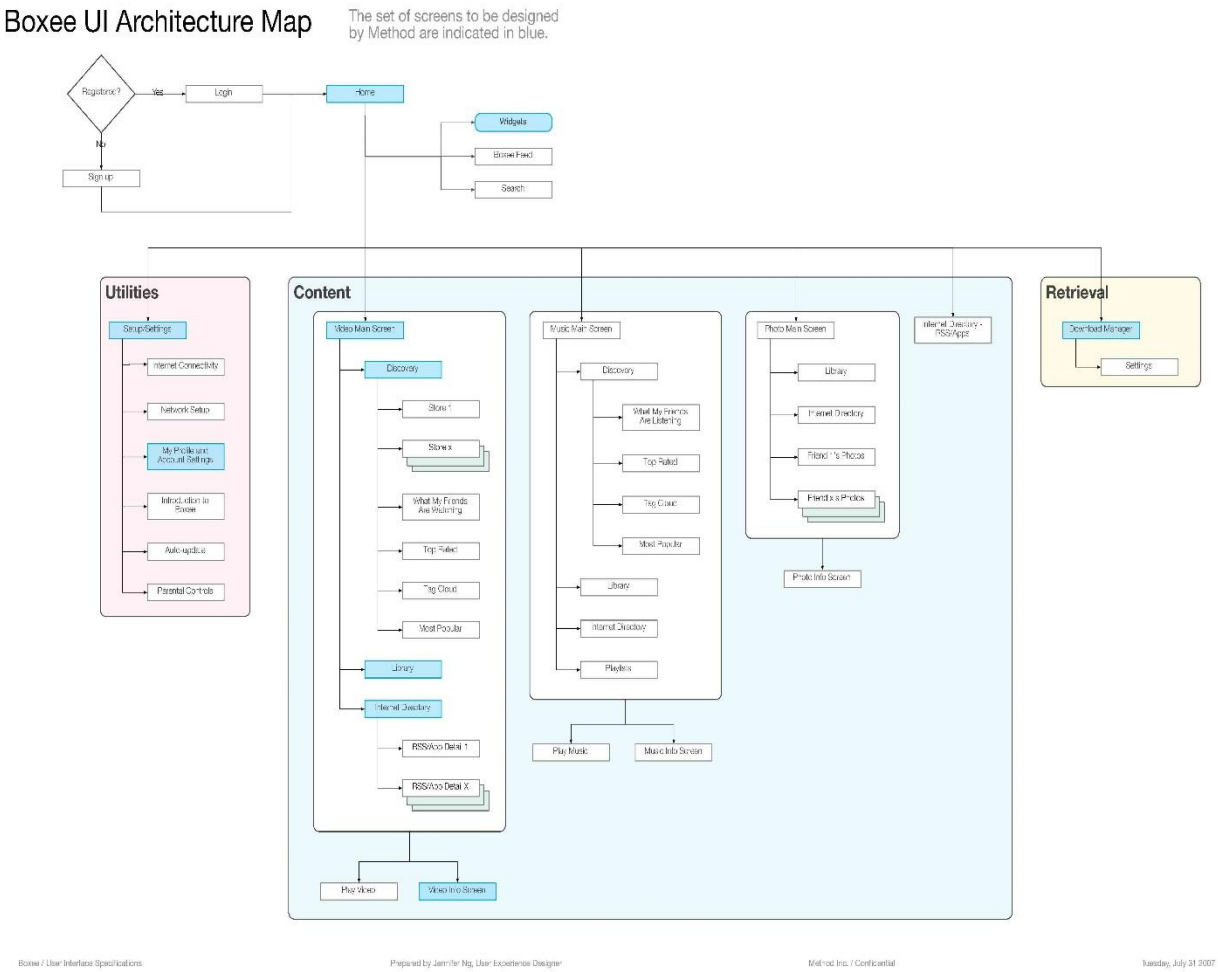
1). XBMC's own internal cross-platform video and audio players can not play any audio or video files that are protected/encrypted with DRM.

2). It does not support some of the audio and video extensions.

3). TV-tuner support requires use of a third-party plugin.

4). XBMC does not currently support binary addons from third-party developers.

2.1.2) Boxee Media Center:



Boxee is a cross-platform freeware HTPC (Home Theater PC) software application with a 10-foot user interface and social networking features designed for the living-room TV that enables its users to view, rate and recommend content to their friends through many social network services and interactive media related

features. Boxee is a media center software which uses XMBC as an application framework for its GUI and media player core platform, together with some custom and proprietary additions.

Features:

1). Audio/video playback and handling.

2). Boxee AppBox Add-on Store and Plugin Apps.

3). Social Networking Layers.

Limitations:

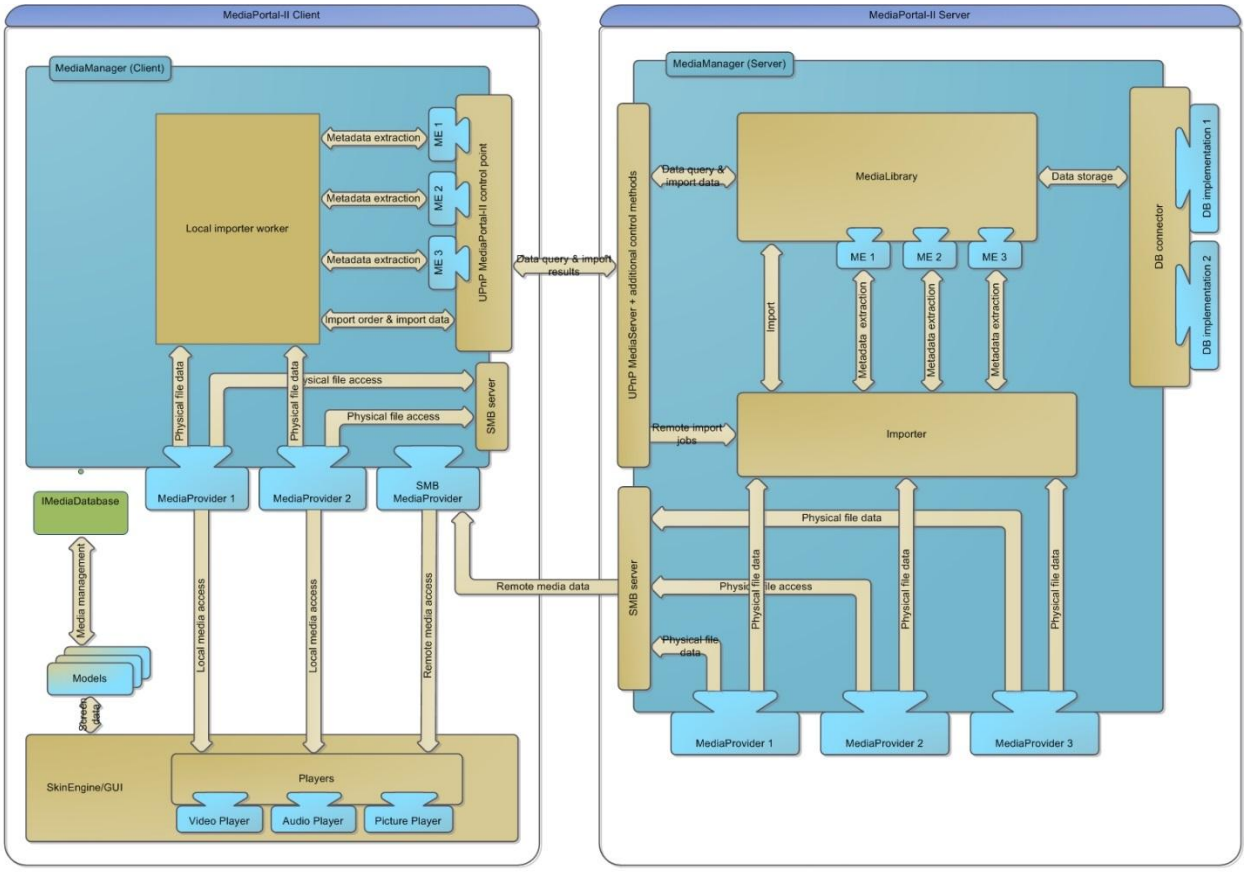
1). Boxee Box does not support Flash Player 11.

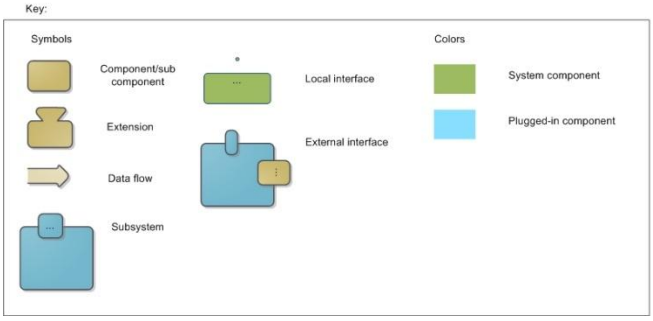
2). Boxee cannot play any audio/music files protected/encrypted with Digital rights management (DRM).

3). Boxee is currently available for x86-based platforms, and a x86-64 version is available for Windows. Boxee is not yet available for ARM, PowerPC, or MIPS processor architectures.

4). Boxee requires a DirectX 9.0 or OpenGL 1.4 with GLSL (or newer) hardware accelerated graphics GPU and matching device drivers.

2.1.3) Media Portal





MediaPortal is an open-source media center (HTPC) software project, often considered an alternative to Windows Media Center. It provides a10-foot user interface for performing typical PVR/TiVo functionality, including playing, pausing, and recording live TV; playing DVDs, videos, and music; viewing pictures; and other functions. Plugins allow it to perform additional tasks, such as watching online video, listening to music from online services such as Last.fm, and launching other applications such as games.

Features:

1). Music, Video and Image playback and handling.

2). Recording, pause and time shifting of TV and Radio broadcasts.

3). Integrated weather forecasts.

4) .Built-in RSS Reader.

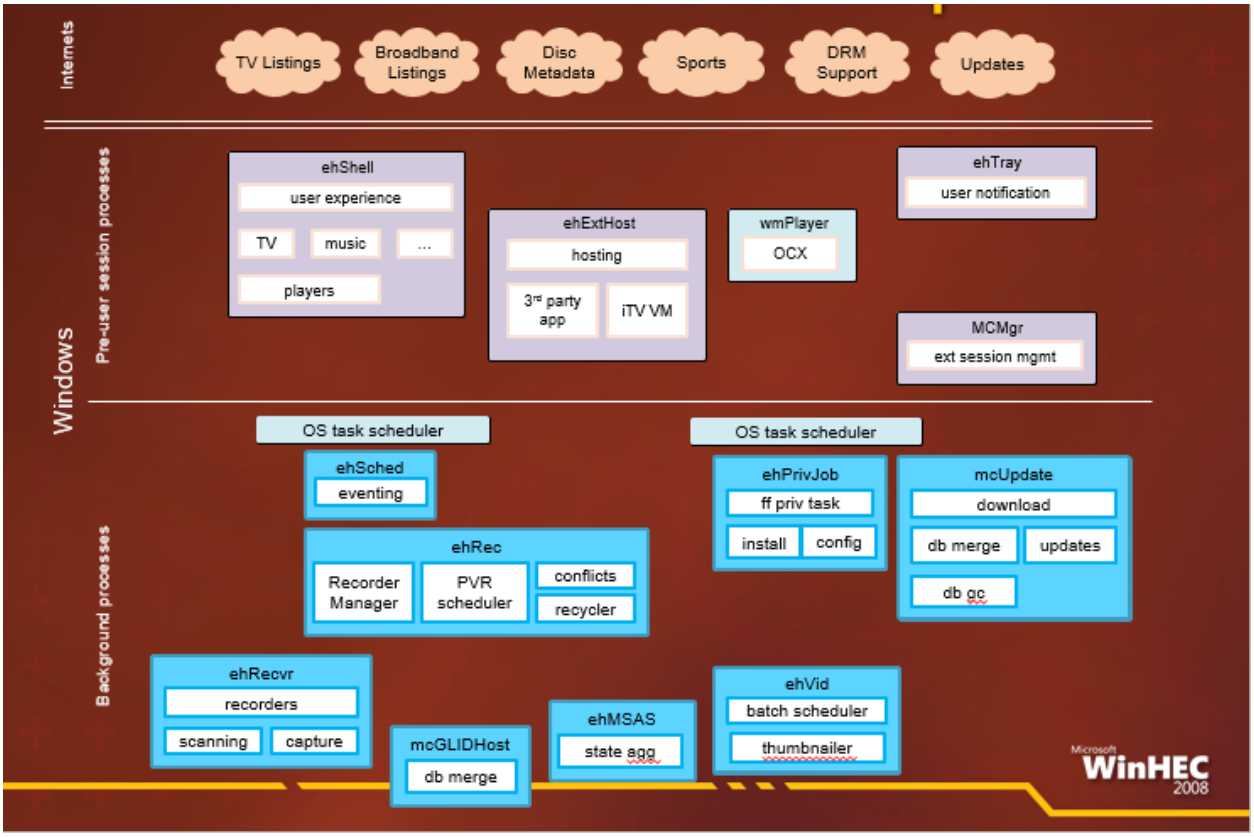
5). Supports all the device controls supported by windows.

Limitations:

1). Designed only for Windows OS.

2). Lacks support of many audio and video extensions.

2.1.4) Windows Media Center



Windows Media Center is a digital video recorder and media player developed by Microsoft. It is an application that allows users to view and record live television, as well as organize and play music and videos. The application is included in various versions of Windows like, Windows Vista Home Premium and

Ultimate, and all editions of Windows 7 except for Starter and Home Basic. It is also available for Windows 8 Pro as an add-on.

Windows Media Center can play slideshows, videos and music from local hard drives, optical drives and network locations. Users can streamtelevision programs and films through selected services such as Netflix. Content can be played back on computer monitors or ontelevision sets through the use of devices called Windows Media Center Extenders.

Features:

1). Video, Audio and picture handling and Playback.

2). Windows Media Center organizes and displays videos and music found on both local and networked computers.

3). Windows Media Center allows synchronization with certain portable devices. These devices include Windows Mobile Pocket PCs, smartphones, Portable Media Centers and other players that can sync with Windows Media Player.

Limitations:

1). Lacks support of many audio and video extensions.

2). Poor Performance in High Definition Video Playback.

3). Designed only for Windows.

Qt

2.2.1) What is Qt?

Qt is a full development framework with tools designed to streamline the creation of applications and user interfaces for desktop, embedded and mobile platforms.

Qt Framework contains intuitive C++ API along with QML (CSS/Javascript – like programming) for rapid UI creation.

Qt is a cross-platform application framework that is widely used for developing application software with a graphical user interface (GUI) (in which cases Qt is classified as a *widget toolkit*), and also used for developing non-GUI programs such as command-line tools and consoles for servers.

With Qt, you can reuse code efficiently to target multiple platforms with one code base. The modular C++ class library and developer tools easily enables developers to create applications for one platform and easily build and run to deploy on another platform.

Qt uses standard C++ but makes extensive use of a special code generator(called the *Meta Object Compiler*, or *moc*) together with several macros to enrich the language. Qt can also be used in several other programming languages via language bindings. It runs on the major desktop platforms and some of the mobile platforms. It has extensive internationalization support. Non-GUI features include SQL database access, XML parsing, thread management, network support, and a unified cross-platform application programming interface (API) for file handling.

2.2.2) About Qt:

Original Author : Haavard Nord and Eirik Chambe-Eng

Developer : Qt Project, Digia

Initial Release : 20 May 1995

Stable Release : August 28, 2013; version 5.1.1

Native Language : C++

Platform : Cross-Platform (Windows, Macintosh, Linux, Symbian etc.)

License : LGPL (Open-Source Version), Qt Commercial License (Commercial Qt License)

2.2.3) Applications:

VLC Media Player (Developed by Video Lan Project, 2001)

KDE (Founder – Matthias Ettrech, 1996)

Symbian OS (Developed by Accenture, 1997)

Skype (Developed by Microsoft Skype Division, 2003)

Google Earth (Developed by Google Inc., 2001)

Adobe Photoshop (Developed by Adobe Systems, 1990)

Amazon Kindle (Developed by Amazon, 2007)

AutoDesk Maya (Developed by Autodesk Inc., 1998)

Blackberry (Developed by Blackberry)

2.2.4) Literature Review Qt:

The Qt Framework was initially developed by Haavard Nord and Eirik Chambe-Eng. They were working on a C++ Database application for ultrasound images that needed to be able to run with a GUI on Unix, Macintosh and Windows. So, they decided to make a object-oriented display system later which came out to be a cross-platform GUI framework.

In 1991, Haavard started writing classes for GUI and collaborated with Erik who came up with the idea of “signals and slots” to make a powerful GUI.

In 1993, Haavard and Erik had developed Qt’s first graphics Kernel and they implemented their own widgets.

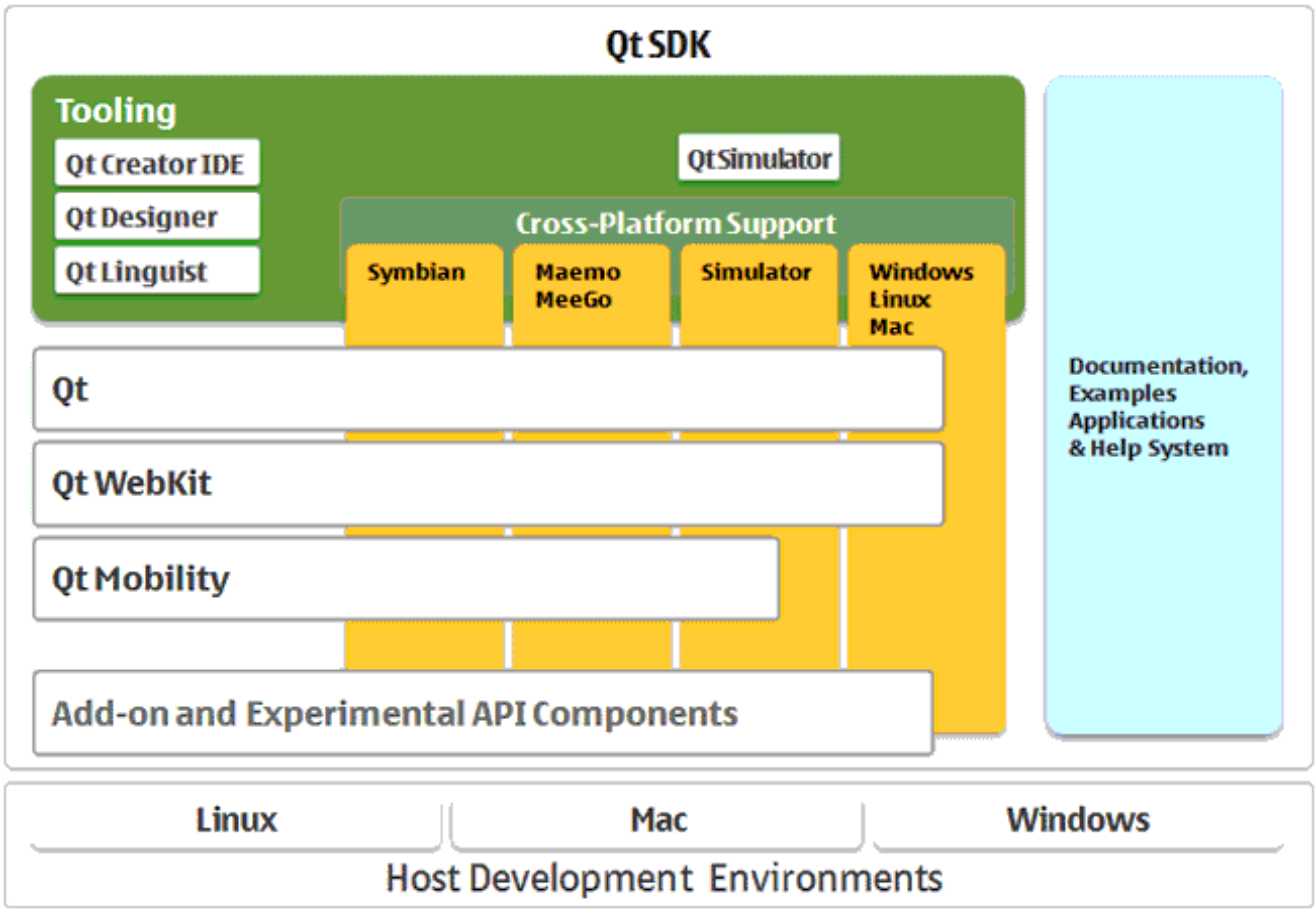
In 1995, these two came up with the name Qt for the framework and company was incorporated as TrollTech in 1995. The first version of Qt was released.

In 2008, Nokia acquired Trolltech and focused on the development for Symbian phones and extended the Qt’s port to Symbian as well.

In September 2012, Digia acquired Qt and focused on further porting it to more platforms.

Currently, the framework is under development to be ported in Android and iOS as well.

2.2.5) Architecture Of Qt:



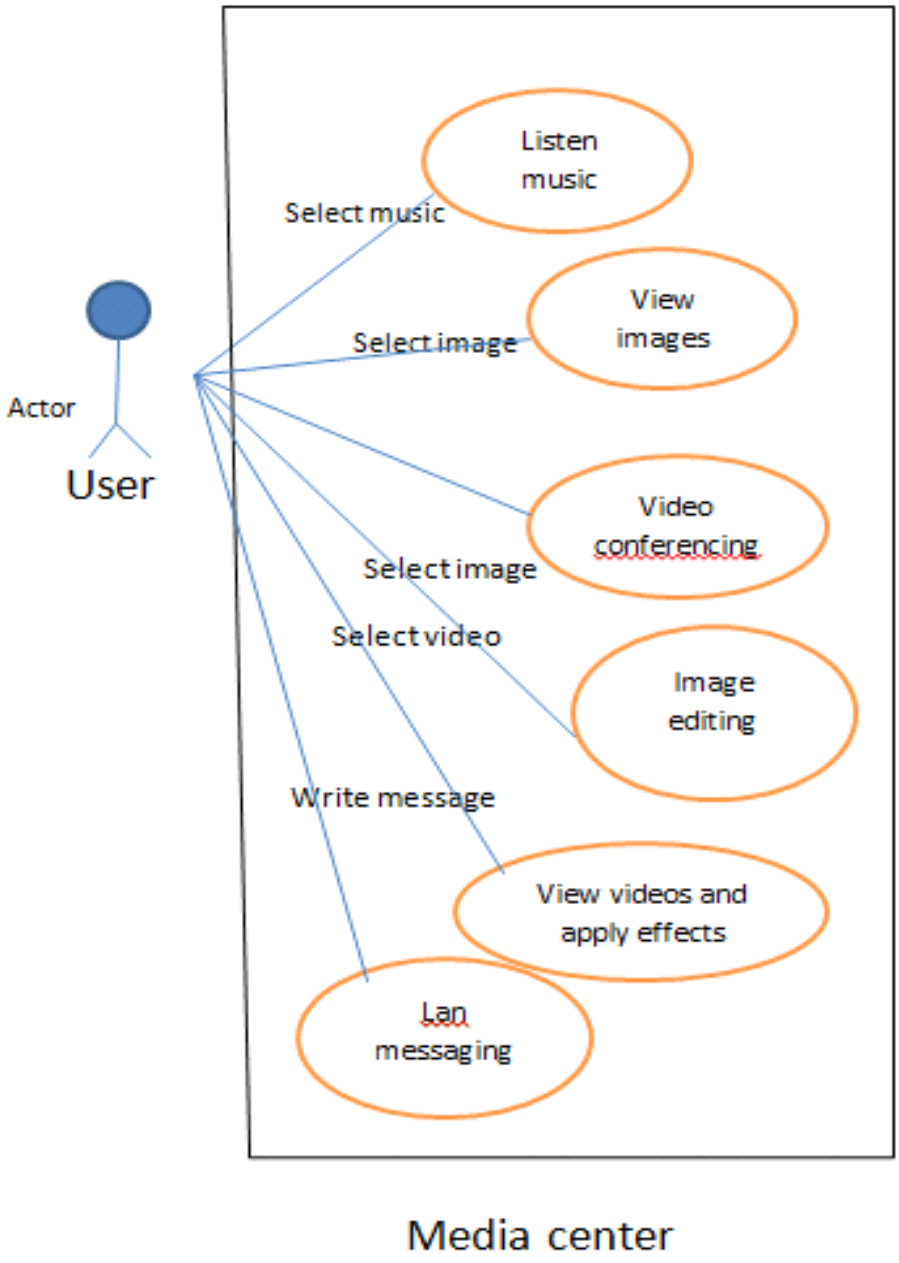
Chapter – 3

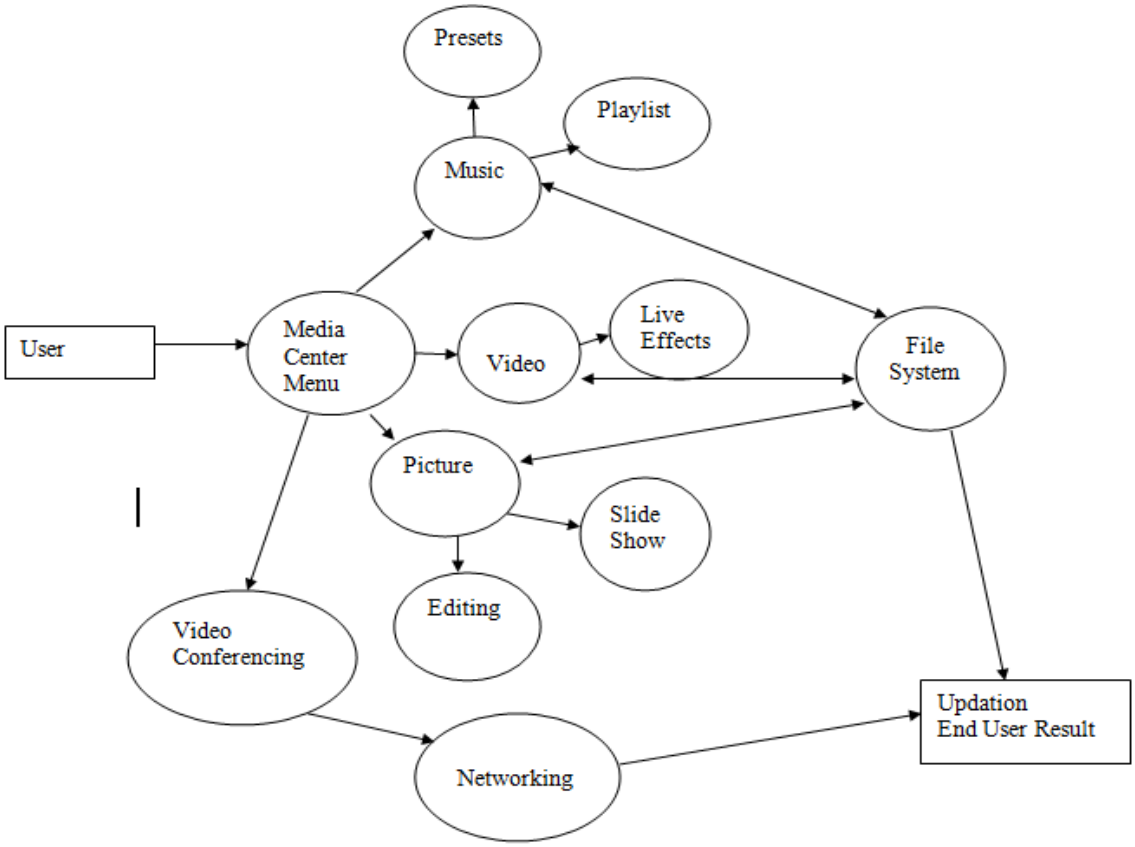
Implementation

Cross Platform Media Center

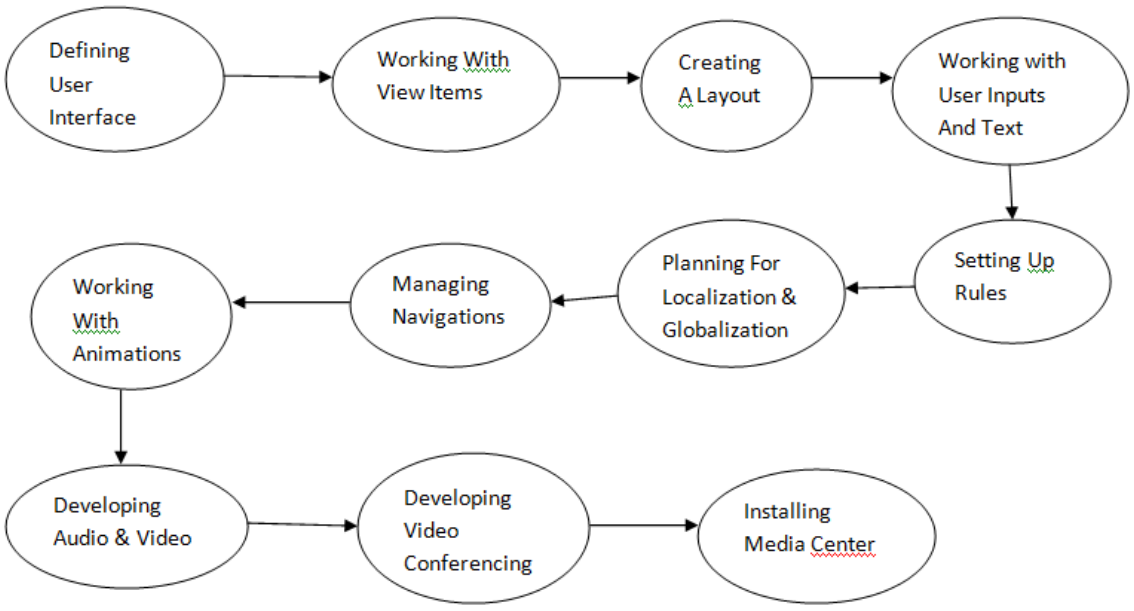
Design of Media Center

Use Case Diagram





Level 2 DFD



Process Flow

Modules

3.2.1) Audio Module:

Audio player allows us to play music files and provides simple playback control, such as pausing, stopping, and resuming the music.

The player has a button group with the play, pause, and stop buttons familiar from most music players. The top-most slider controls the position in the media stream, and the bottom slider allows adjusting the sound volume.

The user can use a file dialog to add music files to a table, which displays meta information about the music - such as the title, album, and artist. Each row contains information about a single music file; to play it, the user selects that row and presses the play button. Also, when a row is selected, the files in the table are queued for playback.

Phonon offers playback of sound using an available audio device, e.g., a sound card or an USB headset. For the implementation, we use two objects: a Media Object, which controls the playback, and an Audio Output, which can output the audio to a sound device. I will explain how they cooperate when we encounter them in the code. For a high-level introduction to Phonon, see its overview.

The API of Phonon is implemented through an intermediate technology on each supported platform: DirectShow, QuickTime, and GStreamer. The sound formats supported may therefore vary from system to system.

My player consists of one class, MainWindow, which both constructs the GUI and handles the playback. We will now go through the parts of its definition and implementation that concerns Phonon.

Code :

MainWindow Class Defination
mainwindow.h

```
#ifndefMAINWINDOW_H
#defineMAINWINDOW_H
#include<QMainWindow>
#include<phonon/audiooutput.h>
#include<phonon/seekslider.h>
#include<phonon/mediaobject.h>
#include<phonon/volumeslider.h>
#include<phonon/backendcapabilities.h>
#include<QList>
classQAction;
classQTableWidget;
classQLCDNumber;
classMainWindow:publicQMainWindow
{
Q_OBJECT
public:
MainWindow();
QSize sizeHint()const{
returnQSize(500,300);
}
privateslots:
voidaddFiles();
voidabout();
voidstateChanged(Phonon::StatenewState,Phonon::StateoldState);
voidtick(qint64time);
voidsourceChanged(constPhonon::MediaSource&source);
voidmetaStateChanged(Phonon::StatenewState,Phonon::StateoldState);
voidaboutToFinish();
voidtableClicked(introw,intcolumn);
private:
voidsetupActions();
voidsetupMenus();
voidsetupUi();
Phonon::SeekSlider*seekSlider;
Phonon::MediaObject*mediaObject;
Phonon::MediaObject*metaInformationResolver;
Phonon::AudioOutput*audioOutput;
Phonon::VolumeSlider*volumeSlider;
QList<Phonon::MediaSource>sources;
QAction*playAction;
```

```
QAction*pauseAction;
QAction*stopAction;
QAction*nextAction;
QAction*previousAction;
QAction*addFilesAction;
QAction*exitAction;
QAction*aboutAction;
QAction*aboutQtAction;
QLCDNumber*timeLcd;
QTableWidget*musicTable;
};
#endif
```

MainWindow Class Implementation
(mainwindow.cpp)

```
#include<QtGui>
#include"mainwindow.h"
MainWindow::MainWindow()
{
audioOutput=newPhonon::AudioOutput(Phonon::MusicCategory,this);
mediaObject=newPhonon::MediaObject(this);
metaInformationResolver=newPhonon::MediaObject(this);
mediaObject->setTickInterval(1000);
connect(mediaObject,SIGNAL(tick(qint64)),this,SLOT(tick(qint64)));
connect(mediaObject,SIGNAL(stateChanged(Phonon::State,Phonon::State)),
this,SLOT(stateChanged(Phonon::State,Phonon::State)));
connect(metaInformationResolver,SIGNAL(stateChanged(Phonon::State,Phonon::State)),
this,SLOT(metaStateChanged(Phonon::State,Phonon::State)));
connect(mediaObject,SIGNAL(currentSourceChanged(Phonon::MediaSource)),
this,SLOT(sourceChanged(Phonon::MediaSource)));
connect(mediaObject,SIGNAL(aboutToFinish()),this,SLOT(aboutToFinish()));
Phonon::createPath(mediaObject,audioOutput);
setupActions();
setupMenus();
setupUi();
timeLcd->display("00:00");
}
voidMainWindow::addFiles()
{
QStringListfiles=QFileDialog::getOpenFileNames(this,tr("SelectMusicFiles"),
QDesktopServices::storageLocation(QDesktopServices::MusicLocation));
if(files.isEmpty())
```

```
return;
intindex=sources.size();
foreach(QStringstring,files){
Phonon::MediaSourcesource(string);
sources.append(source);
}
if(!sources.isEmpty())
metaInformationResolver->setCurrentSource(sources.at(index));
}
voidMainWindow::about()
{
QMessageBox::information(this,tr("AboutMusicPlayer"),
tr("TheMusicPlayerexampleshowshowtousePhonon-themultimedia"
"frameworkthatcomeswithQt-tocreateasimplemusicplayer."));
}
voidMainWindow::stateChanged(Phonon::StatenewState,Phonon::State/*oldState*/)
{
switch(newState){
casePhonon::ErrorState:
if(mediaObject->errorType()==Phonon::FatalError){
QMessageBox::warning(this,tr("FatalError"),
mediaObject->errorString());
}else{
QMessageBox::warning(this,tr("Error"),
mediaObject->errorString());
}
break;
casePhonon::PlayingState:
playAction->setEnabled(false);
pauseAction->setEnabled(true);
stopAction->setEnabled(true);
break;
casePhonon::StoppedState:
stopAction->setEnabled(false);
playAction->setEnabled(true);
pauseAction->setEnabled(false);
timeLcd->display("00:00");
break;
casePhonon::PausedState:
pauseAction->setEnabled(false);
stopAction->setEnabled(true);
playAction->setEnabled(true);
break;
casePhonon::BufferingState:
```

```
break;
default:
;
}
}
voidMainWindow::tick(qint64time)
{
QTimedisplayTime(0,(time/60000)%60,(time/1000)%60);
timeLcd->display(displayTime.toString("mm:ss"));
}
voidMainWindow::tableClicked(introw,int/*column*/)
{
boolwasPlaying=mediaObject->state()==Phonon::PlayingState;
mediaObject->stop();
mediaObject->clearQueue();
if(row>=sources.size())
return;
mediaObject->setCurrentSource(sources[row]);
if(wasPlaying)
mediaObject->play();
else
mediaObject->stop();
}
voidMainWindow::sourceChanged(constPhonon::MediaSource&source)
{
musicTable->selectRow(sources.indexOf(source));
timeLcd->display("00:00");
}
voidMainWindow::metaStateChanged(Phonon::StatenewState,Phonon::State/*oldState*/)
{
if(newState==Phonon::ErrorState){
QMessageBox::warning(this,tr("Erroropeningfiles"),
metaInformationResolver->errorString());
while(!sources.isEmpty()&&
!(sources.takeLast()==metaInformationResolver->currentSource())){}/*loop*/;
return;
}
if(newState!=Phonon::StoppedState&&newState!=Phonon::PausedState)
return;
if(metaInformationResolver->currentSource().type()==Phonon::MediaSource::Invalid)
return;
QMap<QString,QString>metaData=metaInformationResolver->metaData();
QStringtitle=metaData.value("TITLE");
if(title=="")
```

```
title=metaInformationResolver->currentSource().fileName();
QTableWidgetItem*titleItem=newQTableWidgetItem(title);
titleItem->setFlags(titleItem->flags()^Qt::ItemIsEditable);
QTableWidgetItem*artistItem=newQTableWidgetItem(metaData.value("ARTIST"));
artistItem->setFlags(artistItem->flags()^Qt::ItemIsEditable);
QTableWidgetItem*albumItem=newQTableWidgetItem(metaData.value("ALBUM"));
albumItem->setFlags(albumItem->flags()^Qt::ItemIsEditable);
QTableWidgetItem*yearItem=newQTableWidgetItem(metaData.value("DATE"));
yearItem->setFlags(yearItem->flags()^Qt::ItemIsEditable);
intcurrentRow=musicTable->rowCount();
musicTable->insertRow(currentRow);
musicTable->setItem(currentRow,0,titleItem);
musicTable->setItem(currentRow,1,artistItem);
musicTable->setItem(currentRow,2,albumItem);
musicTable->setItem(currentRow,3,yearItem);
if(musicTable->selectedItems().isEmpty()){
musicTable->selectRow(0);
mediaObject->setCurrentSource(metaInformationResolver->currentSource());
}
Phonon::MediaSourcesource=metaInformationResolver->currentSource();
intindex=sources.indexOf(metaInformationResolver->currentSource())+1;
if(sources.size()>index){
metaInformationResolver->setCurrentSource(sources.at(index));
}
else{
musicTable->resizeColumnsToContents();
if(musicTable->columnWidth(0)>800)
musicTable->setColumnWidth(0,800);
}
}
voidMainWindow::aboutToFinish()
{
intindex=sources.indexOf(mediaObject->currentSource())+1;
if(sources.size()>index){
mediaObject->enqueue(sources.at(index));
}
}
voidMainWindow::setupActions()
{
playAction=newQAction(style()->standardIcon(QStyle::SP_MediaPlay),tr("Play"),this);
playAction->setShortcut(tr("Ctrl+P"));
playAction->setDisabled(true);
pauseAction=newQAction(style()->standardIcon(QStyle::SP_MediaPause),tr("Pause"),this);
pauseAction->setShortcut(tr("Ctrl+A"));
```

```
pauseAction->setDisabled(true);
stopAction=newQAction(style()->standardIcon(QStyle::SP_MediaStop),tr("Stop"),this);
stopAction->setShortcut(tr("Ctrl+S"));
stopAction->setDisabled(true);
nextAction=newQAction(style()->standardIcon(QStyle::SP_MediaSkipForward),tr("Next"),this);
nextAction->setShortcut(tr("Ctrl+N"));
previousAction=newQAction(style()-
>standardIcon(QStyle::SP_MediaSkipBackward),tr("Previous"),this);
previousAction->setShortcut(tr("Ctrl+R"));
addFilesAction=newQAction(tr("Add&Files"),this);
addFilesAction->setShortcut(tr("Ctrl+F"));
exitAction=newQAction(tr("E&xit"),this);
exitAction->setShortcuts(QKeySequence::Quit);
aboutAction=newQAction(tr("A&bout"),this);
aboutAction->setShortcut(tr("Ctrl+B"));
aboutQtAction=newQAction(tr("About&Qt"),this);
aboutQtAction->setShortcut(tr("Ctrl+Q"));
connect(playAction,SIGNAL(triggered()),mediaObject,SLOT(play()));
connect(pauseAction,SIGNAL(triggered()),mediaObject,SLOT(pause()));
connect(stopAction,SIGNAL(triggered()),mediaObject,SLOT(stop()));
connect(addFilesAction,SIGNAL(triggered()),this,SLOT(addFiles()));
connect(exitAction,SIGNAL(triggered()),this,SLOT(close()));
connect(aboutAction,SIGNAL(triggered()),this,SLOT(about()));
connect(aboutQtAction,SIGNAL(triggered()),qApp,SLOT(aboutQt()));
}
voidMainWindow::setupMenus()
{
QMenu*fileMenu=menuBar()->addMenu(tr("&File"));
fileMenu->addAction(addFilesAction);
fileMenu->addSeparator();
fileMenu->addAction(exitAction);
QMenu*aboutMenu=menuBar()->addMenu(tr("&Help"));
aboutMenu->addAction(aboutAction);
aboutMenu->addAction(aboutQtAction);
}
voidMainWindow::setupUi()
{
QToolBar*bar=newQToolBar;
bar->addAction(playAction);
bar->addAction(pauseAction);
bar->addAction(stopAction);
seekSlider=newPhonon::SeekSlider(this);
seekSlider->setMediaObject(mediaObject);
volumeSlider=newPhonon::VolumeSlider(this);
```

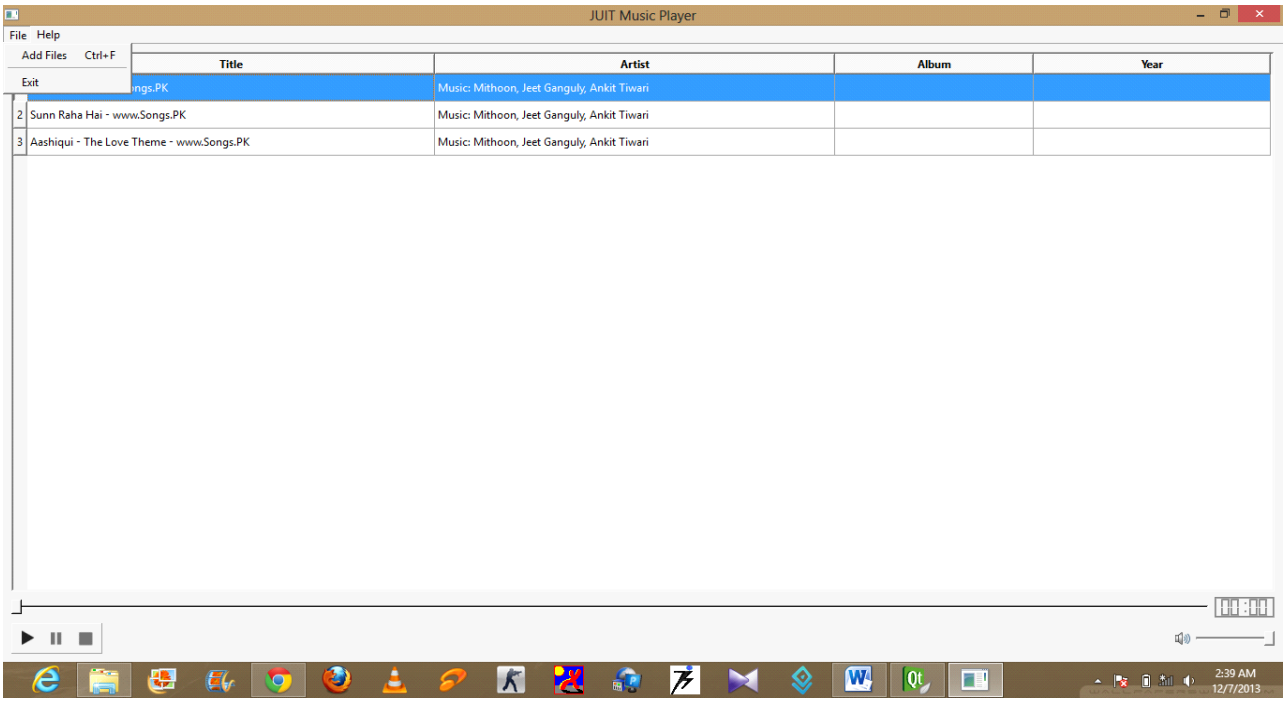
```
volumeSlider->setAudioOutput(audioOutput);
volumeSlider->setSizePolicy(QSizePolicy::Maximum,QSizePolicy::Maximum);
QLabel*volumeLabel=newQLabel;
volumeLabel->setPixmap(QPixmap("images/volume.png"));
QPalettepalette;
palette.setBrush(QPalette::Light,Qt::darkGray);
timeLcd=newQLCDNumber;
timeLcd->setPalette(palette);
QStringListheaders;
headers<<tr("Title")<<tr("Artist")<<tr("Album")<<tr("Year");
musicTable=newQTableWidget(0,4);
musicTable->setHorizontalHeaderLabels(headers);
musicTable->setSelectionMode(QAbstractItemView::SingleSelection);
musicTable->setSelectionBehavior(QAbstractItemView::SelectRows);
connect(musicTable,SIGNAL(cellPressed(int,int)),
this,SLOT(tableClicked(int,int)));
QHBoxLayout*seekerLayout=newQHBoxLayout;
seekerLayout->addWidget(seekSlider);
seekerLayout->addWidget(timeLcd);
QHBoxLayout*playbackLayout=newQHBoxLayout;
playbackLayout->addWidget(bar);
playbackLayout->addStretch();
playbackLayout->addWidget(volumeLabel);
playbackLayout->addWidget(volumeSlider);
QVBoxLayout*mainLayout=newQVBoxLayout;
mainLayout->addWidget(musicTable);
mainLayout->addLayout(seekerLayout);
mainLayout->addLayout(playbackLayout);
QWidget*widget=newQWidget;
widget->setLayout(mainLayout);
setCentralWidget(widget);
setWindowTitle("JUITMusicPlayer");
}
```

main.cpp Code:

```
#include<QtGui>
#include"mainwindow.h"
intmain(intargv,char**args)
{
QApplicationapp(argv,args);
app.setApplicationName("RockMusicPlayer");
app.setQuitOnLastWindowClosed(true);
MainWindowwindow;
#ifdefined(Q_OS_SYMBIAN)
```

```
window.showMaximized();
#else
window.showMaximized();
#endif
returnapp.exec();
}
```

Adding Files To Media Library



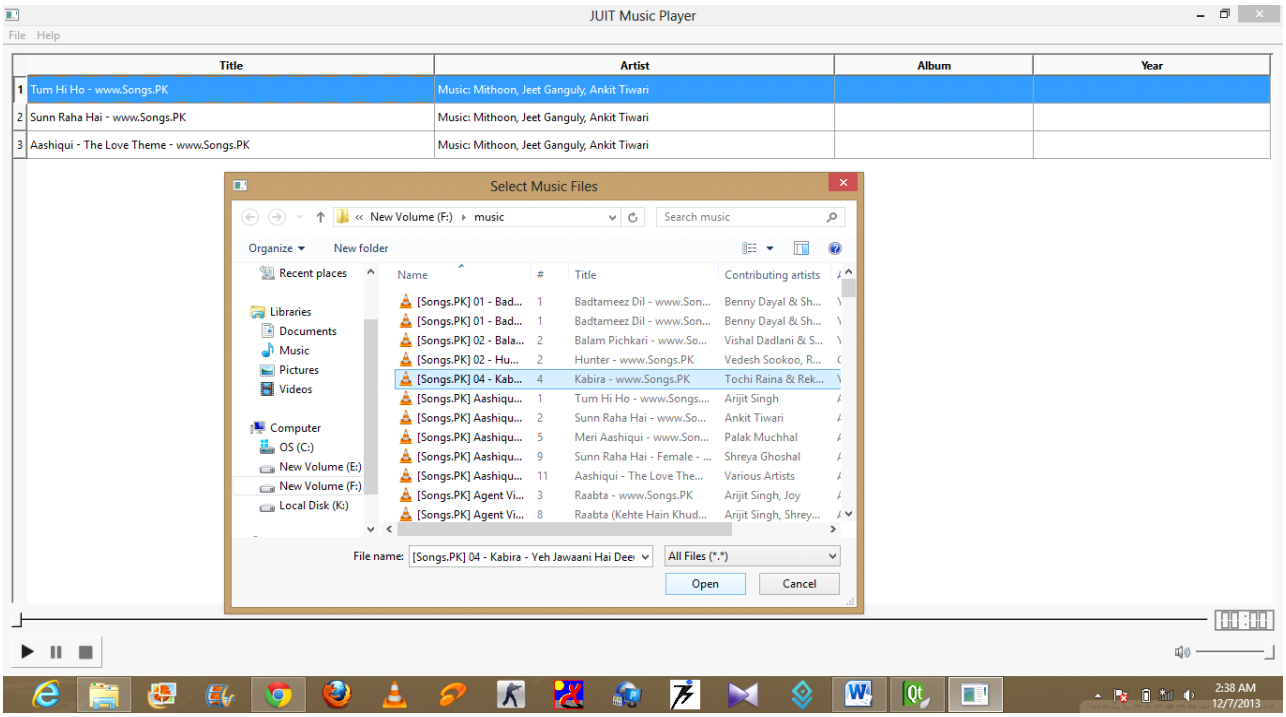

JUIT Music Player
File Help
Title
Artist
Album
Year
1 Tum Hi Ho - www.Songs.PK
Music: Mithoon, Jeet Ganguly, Ankit Tiwari
2 Sunn Raha Hai - www.Songs.PK
Music: Mithoon, Jeet Ganguly, Ankit Tiwari
3 Aashiqui - The Love Theme - www.Songs.PK
Music: Mithoon, Jeet Ganguly, Ankit Tiwari
Select Music Files
New Volume (F:) ▸ music
Search music
Organize
New folder
Recent places
Libraries
Documents
Music
Pictures
Videos
Computer
OS (C:)
New Volume (E:)
New Volume (F:)
Local Disk (K:)
Name
#
Title
Contributing artists
[Songs.PK] 01 - Bad... 1 Badtameez Dil - www.Son... Benny Dayal & Sh...
[Songs.PK] 01 - Bad... 1 Badtameez Dil - www.Son... Benny Dayal & Sh...
[Songs.PK] 02 - Bala... 2 Balam Pichkari - www.So... Vishal Dadlani & S...
[Songs.PK] 02 - Hu... 2 Hunter - www.Songs.PK Vedesh Sookoo, R...
[Songs.PK] 04 - Kab... 4 Kabira - www.Songs.PK Tochi Raina & Rek...
[Songs.PK] Aashiqu... 1 Tum Hi Ho - www.Songs.... Arijit Singh
[Songs.PK] Aashiqu... 2 Sunn Raha Hai - www.So... Ankit Tiwari
[Songs.PK] Aashiqu... 5 Meri Aashiqui - www.Son... Palak Muchhal
[Songs.PK] Aashiqu... 9 Sunn Raha Hai - Female - ... Shreya Ghoshal
[Songs.PK] Aashiqu... 11 Aashiqui - The Love The... Various Artists
[Songs.PK] Agent Vi... 3 Raabta - www.Songs.PK Arijit Singh, Joy
[Songs.PK] Agent Vi... 8 Raabta (Kehte Hain Khud... Arijit Singh, Shrey...
File name: [Songs.PK] 04 - Kabira - Yeh Jawaani Hai Dee
All Files (*.*)
Open
Cancel
2:38 AM
12/7/2013

Image Module :

Fucntionality :

Open: This function is used to open an image file of every format. i.e jpg, jpeg, bmp, raw, png etc.

Print: This function is used to print the already open image with the help of existing printer facility provided by windows.

Zoom In: Used to zoom in the image opened by 25 % .

Zoom Out: Used to zoom out the image opened by 25% .

Blur Image: Used To blur the image opened and not changing the actual copy but opening it in a new window.

Colorize Blue: The whole image will appear blue with the help of this function and the actual copy is not harmed while the applied effect copy is opened in a new Window

Colorize Red: Same function as colorize blue but the colorizing colour is red.
Drop Shadow: It will provide a shadow to the image again by not changing the actual copy and opening the effect in a new Window.

Transparent: It will make the image transparent with respect to window. Effect is applied in a new Window.

Normal Size: It will open the image in its predefined pixel size.

Fit To Window: It will resize the image such that it will fit to window according to the screen size.

Help Function is also provided in which we have explained about Qt and about our application.

Imageviewer.cpp Code:

```
#include <QtGui>
#include "imageviewer.h"
#include "secdialog.h"
ImageViewer::ImageViewer()
{
menuBar=new QMenuBar(this);
mainLayout=new QVBoxLayout;
imageLabel = new QLabel;
imageLabel->setBackgroundRole(QPalette::Base);
imageLabel->setSizePolicy(QSizePolicy::Ignored, QSizePolicy::Ignored);
imageLabel->setScaledContents(true);
//view =new QGraphicsView;
scrollArea = new QScrollArea;
scrollArea->setBackgroundRole(QPalette::Dark);
scrollArea->setWidget(imageLabel);
scrollArea->setAlignment(Qt::AlignCenter);
scrollArea->setMinimumSize(1000,600);
mainLayout->addWidget(menuBar);
mainLayout->setSpacing(0);
mainLayout->addWidget(scrollArea);
mainLayout->setMargin(0);
setLayout(mainLayout);
currFile="";
createActions();
createMenus();
setWindowTitle(tr("Image Viewer"));
resize(500, 400);
}
void ImageViewer::open()
{
   QString fileName = QFileDialog::getOpenFileName(this,
tr("Open File"), QDir::currentPath());
if (!fileName.isEmpty()) {
          QImage image(fileName);
if (image.isNull()) {
             QMessageBox::information(this, tr("Image Viewer"),
tr("Cannot load %1.").arg(fileName));
return;
          }
imageLabel->setPixmap(QPixmap::fromImage(image));
currFile=fileName;
scaleFactor = 1.0;
```

```
printAct->setEnabled(true);
fitToWindowAct->setEnabled(true);
updateActions();
if (!fitToWindowAct->isChecked())
imageLabel->adjustSize();
       }
      }
void ImageViewer::print()
{
   Q_ASSERT(imageLabel->pixmap());
#ifndef QT_NO_PRINTER
   QPrintDialog dialog(&printer, this);
if (dialog.exec()) {
      QPainter painter(&printer);
      QRect rect = painter.viewport();
      QSize size = imageLabel->pixmap()->size();
size.scale(rect.size(), Qt::KeepAspectRatio);
painter.setViewport(rect.x(), rect.y(), size.width(), size.height());
painter.setWindow(imageLabel->pixmap()->rect());
painter.drawPixmap(0, 0, *imageLabel->pixmap());
   }
#endif
}
void ImageViewer::zoomIn()
{
scaleImage(1.25);
}
void ImageViewer::zoomOut()
{
scaleImage(0.75);
}
int ImageViewer::radialGradient()
{
gv=new GraphicsView(this);
   QGraphicsView view;
view.setScene(new QGraphicsScene);
     QGraphicsPixmapItem *p = view.scene()->addPixmap(QPixmap(currFile));
     QGraphicsBlurEffect blur;
blur.setBlurHints(QGraphicsBlurEffect::QualityHint);
blur.setBlurRadius(10); // default value is 5px
p->setGraphicsEffect(&blur);
     QHBoxLayout hb;
hb.addWidget(&view);
gv->showMaximized();
```

```
gv->setLayout(&hb);
return gv->exec();
}
int ImageViewer::colorize()
{
gv=new GraphicsView(this);
   QGraphicsView view;
view.setScene(new QGraphicsScene);
      QSlider *slider=new QSlider(Qt::Horizontal);
slider->setRange(1,10);
      QGraphicsPixmapItem *p = view.scene()->addPixmap(QPixmap(currFile));
      QGraphicsColorizeEffect colorize;
colorize.setColor(QColor(Qt::red)); //Default value is blue
colorize.setStrength(0.5); //Default value is 1.0
p->setGraphicsEffect(&colorize);
      QHBoxLayout hb;
hb.addWidget(&view);
hb.addWidget(slider);
gv->showMaximized();
gv->setLayout(&hb);
return gv->exec();
}
int ImageViewer::colorizeBlue()
{
gv=new GraphicsView(this);
   QGraphicsView view;
view.setScene(new QGraphicsScene);
      QGraphicsPixmapItem *p = view.scene()->addPixmap(QPixmap(currFile));
      QGraphicsColorizeEffect colorize;
colorize.setColor(QColor(Qt::blue)); //Default value is blue
colorize.setStrength(0.5); //Default value is 1.0
p->setGraphicsEffect(&colorize);
      QHBoxLayout hb;
hb.addWidget(&view);
gv->showMaximized();
gv->setLayout(&hb);
return gv->exec();
}
int ImageViewer::colorizeGreen()
{
gv=new GraphicsView(this);
   QGraphicsView view;
view.setScene(new QGraphicsScene);
      QGraphicsPixmapItem *p = view.scene()->addPixmap(QPixmap(currFile));
```

```
      QGraphicsDropShadowEffect effect;
      //effect.setColor(QColor(Qt::blue)); //Default value is blue
effect.setBlurRadius(10); //Default value is 1.0
p->setGraphicsEffect(&effect);
      QHBoxLayout hb;
hb.addWidget(&view);
gv->showMaximized();
gv->setLayout(&hb);
return gv->exec();
}
int ImageViewer::opa()
{
gv=new GraphicsView(this);
   QGraphicsView view;
view.setScene(new QGraphicsScene);
      QGraphicsPixmapItem *p = view.scene()->addPixmap(QPixmap(currFile));
      QGraphicsOpacityEffect effect;
      //effect.setColor(QColor(Qt::blue)); //Default value is blue
effect.setOpacity(0.4);
p->setGraphicsEffect(&effect);
      QHBoxLayout hb;
hb.addWidget(&view);
gv->showMaximized();
gv->setLayout(&hb);
return gv->exec();
}
void ImageViewer::normalSize()
{
imageLabel->adjustSize();
scaleFactor = 1.0;
}
void ImageViewer::fitToWindow()
{
bool fitToWindow = fitToWindowAct->isChecked();
scrollArea->setWidgetResizable(fitToWindow);
if (!fitToWindow) {
normalSize();
   }
updateActions();
}
void ImageViewer::about()
{
   QMessageBox::about(this, tr("About Image Viewer"),
tr("<p>The <b>Image Viewer</b> example shows how to combine QLabel "
```

```
            "and QScrollArea to display an image. QLabel is typically used "
            "for displaying a text, but it can also display an image. "
            "QScrollArea provides a scrolling view around another widget. "
            "If the child widget exceeds the size of the frame, QScrollArea "
            "automatically provides scroll bars. </p><p>The example "
            "demonstrates how QLabel's ability to scale its contents "
            "(QLabel::scaledContents), and QScrollArea's ability to "
            "automatically resize its contents "
            "(QScrollArea::widgetResizable), can be used to implement "
            "zooming and scaling features. </p><p>In addition the example "
            "shows how to use QPainter to print an image.</p>"));
}
void ImageViewer::createActions()
{
openAct = new QAction(tr("&Open..."), this);
openAct->setShortcut(tr("Ctrl+O"));
connect(openAct, SIGNAL(triggered()), this, SLOT(open()));
printAct = new QAction(tr("&Print..."), this);
printAct->setShortcut(tr("Ctrl+P"));
printAct->setEnabled(false);
connect(printAct, SIGNAL(triggered()), this, SLOT(print()));
exitAct = new QAction(tr("E&xit"), this);
exitAct->setShortcut(tr("Ctrl+Q"));
connect(exitAct, SIGNAL(triggered()), this, SLOT(close()));
zoomInAct = new QAction(tr("Zoom &In (25%)"), this);
zoomInAct->setShortcut(tr("Ctrl++"));
zoomInAct->setEnabled(false);
connect(zoomInAct, SIGNAL(triggered()), this, SLOT(zoomIn()));
zoomOutAct = new QAction(tr("Zoom &Out (25%)"), this);
zoomOutAct->setShortcut(tr("Ctrl+-"));
zoomOutAct->setEnabled(false);
connect(zoomOutAct, SIGNAL(triggered()), this, SLOT(zoomOut()));
radialGradientAct= new QAction(tr("Blur Image"),this);
radialGradientAct->setShortcut(tr("Ctrl+r"));
radialGradientAct->setEnabled(true);
connect(radialGradientAct,SIGNAL(triggered()),this,SLOT(radialGradient()));
colorizeAct= new QAction(tr("Colorize Red"),this);
colorizeAct->setShortcut(tr("Ctrl+e"));
colorizeAct->setEnabled(true);
connect(colorizeAct,SIGNAL(triggered()),this,SLOT(colorize()));
colorizeBlueAct= new QAction(tr("Colorize Blue"),this);
colorizeBlueAct->setShortcut(tr("Ctrl+b"));
colorizeBlueAct->setEnabled(true);
connect(colorizeBlueAct,SIGNAL(triggered()),this,SLOT(colorizeBlue()));
```

```
colorizeGreenAct= new QAction(tr("Drop Shadwow"),this);
colorizeGreenAct->setShortcut(tr("Ctrl+d"));
colorizeGreenAct->setEnabled(true);
connect(colorizeGreenAct,SIGNAL(triggered()),this,SLOT(colorizeGreen()));
opacityAct= new QAction(tr("Transparent"),this);
opacityAct->setShortcut(tr("Ctrl+l"));
opacityAct->setEnabled(true);
connect(opacityAct,SIGNAL(triggered()),this,SLOT(opa()));
//dShadowAct= new QAction(tr("Drop Shadow"),this);
//dShadowAct->setShortcut(tr("Ctrl+d"));
//dShadowAct->setEnabled(true);
//connect(dShadowAct(),SIGNAL(triggered()),this,SLOT(dShadow()));
normalSizeAct = new QAction(tr("&Normal Size"), this);
normalSizeAct->setShortcut(tr("Ctrl+S"));
normalSizeAct->setEnabled(false);
connect(normalSizeAct, SIGNAL(triggered()), this, SLOT(normalSize()));
fitToWindowAct = new QAction(tr("&Fit to Window"), this);
fitToWindowAct->setEnabled(false);
fitToWindowAct->setCheckable(true);
fitToWindowAct->setShortcut(tr("Ctrl+F"));
connect(fitToWindowAct, SIGNAL(triggered()), this, SLOT(fitToWindow()));
aboutAct = new QAction(tr("&About"), this);
connect(aboutAct, SIGNAL(triggered()), this, SLOT(about()));
aboutQtAct = new QAction(tr("About &Qt"), this);
connect(aboutQtAct, SIGNAL(triggered()), qApp, SLOT(aboutQt()));
}
void ImageViewer::createMenus()
{
fileMenu = new QMenu(tr("&File"), this);
fileMenu->addAction(openAct);
fileMenu->addAction(printAct);
fileMenu->addSeparator();
fileMenu->addAction(exitAct);
viewMenu = new QMenu(tr("&View"), this);
viewMenu->addAction(zoomInAct);
viewMenu->addAction(zoomOutAct);
viewMenu->addAction(radialGradientAct);
viewMenu->addAction(colorizeAct);
viewMenu->addAction(colorizeBlueAct);
viewMenu->addAction(colorizeGreenAct);
viewMenu->addAction(opacityAct);
viewMenu->addAction(normalSizeAct);
viewMenu->addSeparator();
viewMenu->addAction(fitToWindowAct);
```

```
helpMenu = new QMenu(tr("&Help"), this);
helpMenu->addAction(aboutAct);
helpMenu->addAction(aboutQtAct);
menuBar->addMenu(fileMenu);
menuBar->addMenu(viewMenu);
menuBar->addMenu(helpMenu);
}
void ImageViewer::updateActions()
{
zoomInAct->setEnabled(!fitToWindowAct->isChecked());
zoomOutAct->setEnabled(!fitToWindowAct->isChecked());
normalSizeAct->setEnabled(!fitToWindowAct->isChecked());
}

void ImageViewer::scaleImage(double factor)
{
   Q_ASSERT(imageLabel->pixmap());
scaleFactor *= factor;
imageLabel->resize(scaleFactor * imageLabel->pixmap()->size());
adjustScrollBar(scrollArea->horizontalScrollBar(), factor);
adjustScrollBar(scrollArea->verticalScrollBar(), factor);
zoomInAct->setEnabled(scaleFactor < 3.0);
zoomOutAct->setEnabled(scaleFactor > 0.333);
}
void ImageViewer::adjustScrollBar(QScrollBar *scrollBar, double factor)
{
scrollBar->setValue(int(factor * scrollBar->value()
                        + ((factor - 1) * scrollBar->pageStep()/2)));
}
/*************************************************************************
**/
GraphicsView::GraphicsView(QWidget *parent):QDialog(parent)
{
show();
setAttribute(Qt::WA_QuitOnClose);
}
void GraphicsView::closeEvent(QCloseEvent *event)
{
close();
}
```

Imageviewer.h Code:

```
#ifndef IMAGEVIEWER_H
#define IMAGEVIEWER_H
#include <QMainWindow>
#include <QPrinter>
#include<QPainter>
#include<QRadialGradient>
#include<QGraphicsEllipseItem>
#include <QAbstractGraphicsShapeItem>
#include <QGraphicsPathItem>
#include<QGraphicsPolygonItem>
#include<QGraphicsRectItem>
#include<QDialog>
QT_BEGIN_NAMESPACE
class QAction;
class QLabel;
class QMenu;
class QScrollArea;
class QScrollBar;
class QRadialGradient;
class QPainter;
class QGraphicsPathItem;
class QGraphicsPolygonItem;
class QGraphicsRectItem;
class QAbstractGraphicsShapeItem;
class QGraphicsEllipseItem;
class QGraphicsScene;
class QGraphicsView;
class QGridLayout;
class QVBoxLayout;
QT_END_NAMESPACE
class GraphicsView:public QDialog
{
public:
GraphicsView(QWidget *parent=0);
protected:
void closeEvent(QCloseEvent *event);
};
class ImageViewer : public QWidget
{
   Q_OBJECT
public:
ImageViewer();
```

```
private slots:
void open();
void print();
void zoomIn();
void zoomOut();
int radialGradient();
int colorize();
int colorizeBlue();
int colorizeGreen();
int opa();
void normalSize();
void fitToWindow();
void about();
private:
void createActions();
void createMenus();
void updateActions();
void scaleImage(double factor);
void adjustScrollBar(QScrollBar *scrollBar, double factor);
   QPixmap pixmap;
   QLabel *imageLabel;
   QScrollArea *scrollArea;
double scaleFactor;
#ifndef QT_NO_PRINTER
   QPrinter printer;
#endif
   QAction *openAct;
   QAction *printAct;
   QAction *exitAct;
   QAction *zoomInAct;
   QAction *zoomOutAct;
   QAction *radialGradientAct;
   QAction *colorizeAct;
   QAction *colorizeBlueAct;
   QAction *colorizeGreenAct;
   QAction *opacityAct;
   QAction *normalSizeAct;
   QAction *fitToWindowAct;
   QAction *aboutAct;
   QAction *aboutQtAct;
   QMenu *fileMenu;
   QMenu *viewMenu;
   QMenu *helpMenu;
   GraphicsView *gv;
```

```
    QVBoxLayout *mainLayout;
    QGridLayout *widgetLayout;
    QMenuBar *menuBar;
    QString currFile;
    };
#endif
```

Main.cpp Code:

```
#include <QApplication>
#include<windows.h>
#include "imageviewer.h"
#include<QGraphicsBlurEffect>
#include<QGraphicsScene>
#include<QGraphicsView>
int main(int argc, char *argv[])
{
    QApplication app(argc, argv);
    ImageViewer imageViewer;
#if defined(Q_OS_SYMBIAN)
imageViewer.showMaximized();
#else
imageViewer.showMaximized();
#endif
return app.exec();
}
```

References

- Qt Reference Manual - http://qt-project.org/doc/qt-5.1/qtdoc/reference-overview.html
- C++ GUI Development with Qt4 (TextBook) – Published in 2005 by Jasmin Blanchette
- Qt's discussion Forum - http://qt-project.org/forums
- Architecture Reference Xbmc: *http://static.telematicsfreedom.org*
- Architecture Reference Windows Media Center: Microsoft Win Hec 2008 by Jonathan Hutchinson &Luigi Capriotti (18 September 2008)
- Architecture Reference Qt: http://www.cetoni.de/development/software/qt-framework.html
- Applications of Qt: www.wikipedia.com