Специальность: 09.02.07 Информационные системы и программирование

Проверил:

\_\_\_\_\_\_\_\_\_\_\_ Антонов М.Б.

Дата: «» \_\_\_\_\_ 2020 г.

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики»

(Университет ИТМО)

факультет среднего профессионального образования

Отчет

о Лабораторной работе №17

по теме: Работа с аудио и видео в Qt

по дисциплине: Разработка программных модулей

Выполнил:

студент группы Y2433

\_\_\_\_\_\_\_\_\_\_\_ Сафин Р. М.

Санкт-Петербург 2020

Содержание

[1 Постановка задачи 3](#_Toc41139419)

[1.1 Цель 3](#_Toc41139420)

[1.2 Задачи 3](#_Toc41139421)

[1.3 Задание 3](#_Toc41139422)

[1.4 Общее ограничение на лабораторную работу 3](#_Toc41139423)

[2 Диаграммы](#_Toc41139425) 4

[3 Исходный код](#_Toc41139426) 5

[4 ЛИСТИНГ ВЫполнения программы 1](#_Toc41139427)1

[5 Вывод 12](#_Toc41139428)

1. Постановка задачи
   1. Цель

Познакомиться с основами работы мультимедиа в Qt.

* 1. Задачи

1. изучить основы работы с модулем multimedia;
2. Создать простейшее приложение для воспроизведения видео.
   1. Задание

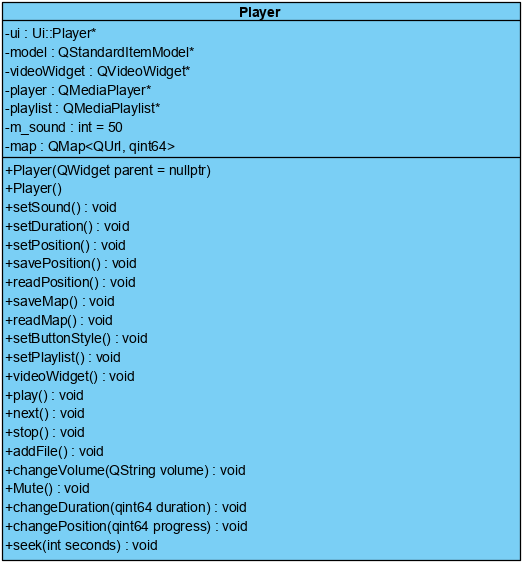
Требуется создать приложение для воспроизведения видеороликов с плейлистом.

Приложение при повторном запуске должно запоминать файлы, которые уже просматривало и воспроизводить их с места остановки.

Программа должна уметь переходить на заданную точку воспроизведения видео и изменять громкость.

* 1. Общее ограничение на лабораторную работу
* использовать разбитие на файлы основной программы (в main.cpp только функция main.cpp);
* документировать весь код, используя аннотации Doxygen.

1. Диаграммы



*Рисунок 1 – Диаграмма классов*

1. Исходный код

Листинг 1 – Код файла main.cpp

/\*\*

\* @file main.cpp

\* @author Roman Safin

\*/

#include "player.h"

#include <QApplication>

int main(int argc, char \*argv[])

{

QApplication a(argc, argv);

Player w;

w.show();

return a.exec();

}

Листинг 2 – Код файла player.h

/\*\*

\* @file player.h

\* @brief audio and video player

\* @author Roman Safin

\*/

#ifndef PLAYER\_H

#define PLAYER\_H

#include <QWidget>

#include <QMediaPlayer>

#include <QMediaPlaylist>

#include <QStandardItemModel>

#include <QVideoWidget>

QT\_BEGIN\_NAMESPACE

namespace **Ui** { class **Player**; }

QT\_END\_NAMESPACE

class **Player** : public QWidget

{

Q\_OBJECT

public:

**Player**(QWidget \*parent = nullptr);

~***Player***();

private:

Ui::Player \*ui;

QStandardItemModel \*model;

QVideoWidget \*widget;

QMediaPlayer \*player;

QMediaPlaylist \*playlist;

int sound = 50;

QMap <QUrl, qint64> map;

private:

/\*\*

\* @brief set sound

\*/

void **setSound**();

/\*\*

\* @brief set duration

\*/

void **setDuration**();

/\*\*

\* @brief save position to map

\*/

void **savePosition**();

/\*\*

\* @brief read position from map

\*/

void **readPosition**();

/\*\*

\* @brief save map to file

\*/

void **saveMap**();

/\*\*

\* @brief read map from file

\*/

void **readMap**();

/\*\*

\* @brief set button style

\*/

void **setButtonStyle**();

/\*\*

\* @brief set playlist

\*/

void **setPlaylist**();

/\*\*

\* @brief set video widget

\*/

void **videoWidget**();

private slots:

/\*\*

\* @brief play file

\*/

void **play**();

/\*\*

\* @brief next file

\*/

void **next**();

/\*\*

\* @brief pause file

\*/

void **pause**();

/\*\*

\* @brief stop file

\*/

void **stop**();

/\*\*

\* @brief add file

\*/

void **addFile**();

/\*\*

\* @brief change volume

\* @param volume

\*/

void **changeVolume**(QString volume);

/\*\*

\* @brief soundMute

\*/

void **Mute**();

/\*\*

\* @brief change duration

\* @param duration

\*/

void **changeDuration**(qint64 duration);

/\*\*

\* @brief positionChanged

\* @param progress

\*/

void **changePosition**(qint64 progress);

/\*\*

\* @brief seek

\* @param seconds

\*/

void **seek**(int seconds);

/\*\*

\* @brief updateDurationInfo

\* @param currentInfo

\*/

};

#endif // PLAYER\_H#endif // PLAYER\_H

Листинг 3 – Код файла player.cpp

/\*\*

\* @file player.cpp

\* @brief audio and video player

\* @author Roman Safin

\*/

#include "player.h"

#include "ui\_player.h"

#include <QFileDialog>

#include <QFileInfo>

#include <QDir>

#include <QTime>

#include <QSpinBox>

#include <QDataStream>

#include <QDebug>

Player::**Player**(QWidget \*parent)

: QWidget(parent)

, ui(new Ui::Player)

{

ui->setupUi(this);

player = new QMediaPlayer(this);

playlist = new QMediaPlaylist(player);

this->readMap();

this->setButtonStyle();

this->setPlaylist();

this->videoWidget();

this->setDuration();

connect(ui->previous\_button, &QToolButton::clicked,playlist, &QMediaPlaylist::previous);

connect(ui->next\_button, &QToolButton::clicked, this, &Player::next);

connect(ui->play\_button, &QToolButton::clicked, this, &Player::play);

connect(ui->pause\_button, &QToolButton::clicked,this, &Player::pause);

connect(ui->stop\_button, &QToolButton::clicked, this, &Player::stop);

connect(ui->add\_button, &QToolButton::clicked, this, &Player::addFile);

}

Player::~***Player***()

{

this->savePosition();

this->saveMap();

delete ui;

delete model;

delete playlist;

delete player;

}

void Player::**setSound**()

{

player->setVolume(sound);

ui->spinBox->setValue(50);

}

void Player::**changeVolume**(QString sound)

{

if(sound.toInt()!=0)

this->sound = sound.toInt();

player->setVolume(sound.toInt());

}

void Player::**Mute**()

{

if(ui->spinBox->value() != 0){

ui->spinBox->setValue(0);

player->setMuted(true);

} else {

ui->spinBox->setValue(this->sound);

player->setMuted(false);

}

}

void Player::**setDuration**()

{

ui->timeSlider->setRange(0, player->duration()/1000);

connect(ui->timeSlider, &QSlider::sliderMoved, this, &Player::seek);

connect(player, &QMediaPlayer::durationChanged, this, &Player::changeDuration);

connect(player, &QMediaPlayer::positionChanged, this, &Player::changePosition);

}

void Player::**changeDuration**(qint64 duration)

{

ui->timeSlider->setMaximum(duration/1000);

}

void Player::**changePosition**(qint64 progress)

{

if(!ui->timeSlider->isSliderDown())

ui->timeSlider->setValue(progress/1000);

}

void Player::**seek**(int seconds)

{

player->setPosition(seconds\*1000);

}

void Player::**savePosition**()

{

map.insert(playlist->currentMedia().request().url(), player->position());

}

void Player::**readPosition**()

{

foreach(QUrl key, map.keys()){

if(key.toString().compare(playlist->currentMedia().request().url().toString()) == 0){

qint64 value = map.value(key);

player->setPosition(value);

}

}

}

void Player::**saveMap**()

{

QFile f ("position.txt");

if (!f.*open*(QIODevice::WriteOnly)) return;

QDataStream str(&f);

str << map;

}

void Player::**readMap**()

{

QFile f ("position.txt");

if (!f.*open*(QIODevice::ReadOnly)) return;

QDataStream str(&f);

map.clear();

str >> map;

f.remove();

}

void Player::**play**()

{

player->play();

ui->play\_button->setEnabled(false);

ui->pause\_button->setEnabled(true);

ui->stop\_button->setEnabled(true);

readPosition();

}

void Player::**next**()

{

savePosition();

playlist->next();

player->play();

}

void Player::**pause**()

{

savePosition();

ui->pause\_button->setEnabled(false);

ui->play\_button->setEnabled(true);

player->pause();

}

void Player::**stop**()

{

ui->stop\_button->setEnabled(false);

ui->play\_button->setEnabled(true);

map.remove(playlist->currentMedia().request().url());

player->stop();

}

void Player::**setButtonStyle**()

{

ui->play\_button->setIcon(style()->*standardIcon*(QStyle::SP\_MediaPlay));

ui->pause\_button->setIcon(style()->*standardIcon*(QStyle::SP\_MediaPause));

ui->next\_button->setIcon(style()->*standardIcon*(QStyle::SP\_MediaSkipForward));

ui->previous\_button->setIcon(style()->*standardIcon*(QStyle::SP\_MediaSkipBackward));

}

void Player::**setPlaylist**()

{

model = new QStandardItemModel(this);

ui->tableView->*setModel*(model);

model->setHorizontalHeaderLabels(QStringList()<< tr("File")

<< tr("File Path"));

ui->tableView->hideColumn(1);

ui->tableView->setSelectionBehavior(QAbstractItemView::SelectRows);

ui->tableView->setSelectionMode(QAbstractItemView::SingleSelection);

ui->tableView->setEditTriggers(QAbstractItemView::NoEditTriggers);

ui->tableView->horizontalHeader()->setStretchLastSection(true);

player->setPlaylist(playlist);

this->setSound();

playlist->setPlaybackMode(QMediaPlaylist::Loop);

connect(ui->tableView, &QTableView::doubleClicked,

[this](const QModelIndex &index){

playlist->setCurrentIndex(index.row());

player->play();

});

connect(playlist, &QMediaPlaylist::currentIndexChanged,

[this](int index){

ui->currentTrack->setText(model->*data*(model->*index*(index, 0)).toString());

});

}

void Player::**videoWidget**()

{

widget = new QVideoWidget(ui->widget);

player->setVideoOutput(widget);

widget->resize(ui->widget->size());

widget->show();

}

void Player::**addFile**()

{

QStringList files = QFileDialog::getOpenFileNames(this, tr("Open files"), QString(), tr("Audio Files (\*.mp3 \*.wav);; ""Video files (\*.avi)"));

foreach (QString filePath, files) {

QList<QStandardItem \*> items;

items.append(new QStandardItem(QDir(filePath).dirName()));

items.append(new QStandardItem(filePath));

model->appendRow(items);

playlist->addMedia(QUrl::fromLocalFile(filePath));

widget = new QVideoWidget;

}}

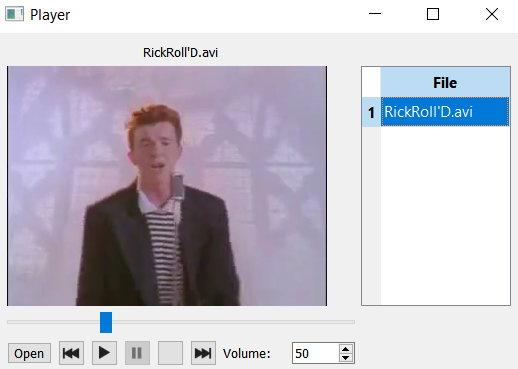
1. ЛИСТИНГ ВЫполнения программы

Рисунок 3 – Интерфейс программы

1. Вывод

В ходе выполнения лабораторной работы был разработан видео/аудиоплеер с возможностями перемотки, паузы, изменения громкости и плейлистом открытых файлов.