

1. Цель и задачи

Цель: познакомится с основами работы мультимедиа в Qt.

Задачи:

* изучить основы работы с модулем multimedia;
* создать простейшее приложение для воспроизведения видео.

1. Задание

Общее ограничение на лабораторную работу:

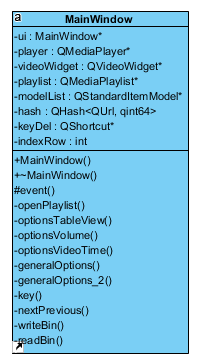
* использовать разбитие на файлы основной программы (в main.cpp только функция main.cpp);
* документировать весь код используя аннотации Doxygen.

Требуется создать приложение для воспроизведение видео роликов с плейлистом.

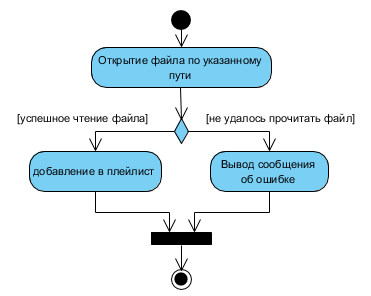
Приложение при повторном запуске должно запоминать файлы, которые уже просматривало и воспроизводить их с места остановки.

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

1. ДИАГРАММЫ



*Рисунок 1 – диаграмма классов*

**

*Рисунок 2 – диаграмма деятельности функции подключения бд*

1. КОд программы

Основной выполняемый файл main.cpp:

#include "mainwindow.h"

#include <QApplication>

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

{

QApplication a(argc, argv);

MainWindow w;

w.show();

return a.exec();

}

Файл mainwindow.cpp:

#include "mainwindow.h"

#include "ui\_mainwindow.h"

MainWindow::MainWindow(QWidget \*parent)

: QMainWindow(parent)

, ui(new Ui::MainWindow)

{

ui->setupUi(this);

modelList = new QStandardItemModel();

ui->tableView->setModel(modelList);

generalOptions();

optionsTableView();

player = new QMediaPlayer;

playlist = new QMediaPlaylist(player);

videoWidget = new QVideoWidget(ui->widget);

player->setVideoOutput(videoWidget);

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

player->setPlaylist(playlist);

connect(ui->pushButton,&QToolButton::clicked,this,&MainWindow::openPlaylist);

connect(ui->tableView, &QTableView::doubleClicked, [this](const QModelIndex &index){playlist->setCurrentIndex(index.row());});

connect(playlist, &QMediaPlaylist::currentIndexChanged, [this](int index){ui->lineEdit->setText(modelList->data(modelList->index(index, 0)).toString());});

connect(ui->tableView, &QTableView::doubleClicked, this,&MainWindow::playVideo);

connect(ui->tabWidget,&QTabWidget::currentChanged,this,&MainWindow::pauseVideo);

connect(ui->playBut, &QToolButton::clicked, player, &QMediaPlayer::play);

connect(ui->pauseBut, &QToolButton::clicked, player, &QMediaPlayer::pause);

optionsVideoTime();

optionsVolume();

key();

connect(ui->info,&QToolButton::clicked,this,&MainWindow::message);

videoWidget->show();

ui->tabWidget->setCurrentIndex(0);

connect(ui->tableView, &QTableView::doubleClicked, this,&MainWindow::findInHash);

connect(ui->playBut, &QToolButton::clicked, this,&MainWindow::startInTab);

nextPrevious();

}

MainWindow::~MainWindow()

{

delete ui;

}

void MainWindow::generalOptions(){

modelList->setHorizontalHeaderLabels(QStringList() << tr("Video Track") << tr("File Path"));

ui->tabWidget->setTabText(0,"Playlist");

ui->tabWidget->setTabText(1,"Video");

ui->lineEdit->setDisabled(1);

ui->playBut->setIcon(QIcon(":/buttons/play"));

ui->pauseBut->setIcon(QIcon(":/buttons/pause"));

this->setWindowIcon(QIcon(":/player/player.png"));

generalOptions\_2();

}

void MainWindow::generalOptions\_2(){

ui->info->setIcon(QIcon(":/buttons/info"));

this->setWindowTitle("Player");

ui->previous->setIcon(QIcon(":/buttons/skip-previous"));

ui->next->setIcon(QIcon(":/buttons/skip-next"));

}

void MainWindow::optionsTableView(){

ui->tableView->hideColumn(1);

ui->tableView->verticalHeader()->setVisible(false);

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

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

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

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

}

void MainWindow::optionsVideoTime(){

connect(player,&QMediaPlayer::durationChanged,ui->videoTime,&QSlider::setMaximum);

connect(player,&QMediaPlayer::positionChanged,ui->videoTime,&QSlider::setValue);

connect(ui->videoTime,&QSlider::sliderReleased,this,&MainWindow::getPosition);

}

void MainWindow::getPosition(){

player->setPosition(ui->videoTime->value());

}

void MainWindow::optionsVolume(){

ui->volume->setValue(100);

connect(ui->volume,&QSlider::valueChanged,player,&QMediaPlayer::setVolume);

ui->label->setText(QString::number(ui->volume->value()));

connect(ui->volume,&QSlider::valueChanged,this,&MainWindow::typeToVolume);

}

void MainWindow::typeToVolume(int value){

ui->label->setText(QString::number(value));

}

void MainWindow::openPlaylist()

{

QStringList files = QFileDialog::getOpenFileNames(this,tr("Open files"),QString(),tr("Video Files (\*.mp4 \*.avi \*.wmv)"));

foreach (QString filePath, files) {

QList<QStandardItem \*> items;

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

items.append(new QStandardItem(filePath));

modelList->appendRow(items);

playlist->addMedia(QUrl("file:///"+filePath));

}

}

void MainWindow::playVideo(){

ui->tabWidget->setCurrentIndex(1);

}

void MainWindow::pauseVideo(int index){

if(index==0){

player->pause();

if(playlist->currentMedia().request().url().toString()!="")

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

player->stop();

}

}

void MainWindow::findInHash(){

QHash <QUrl,qint64>::const\_iterator valueIt=hash.find(playlist->currentMedia().request().url());

if(valueIt != hash.end())

player->setPosition(hash.find(playlist->currentMedia().request().url()).value());

}

bool MainWindow::event(QEvent \*event)

{

if (event->type() == QEvent::Close)

writeBin();

if (event->type() == QEvent::Show)

readBin();

if(event->type()==QEvent::Resize)

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

return QMainWindow::event(event);

}

void MainWindow::writeBin(){

QFile file(QCoreApplication::applicationDirPath()+"/positions.bin");

if (file.open(QIODevice::WriteOnly)){

QDataStream stream(&file);

stream.setVersion(QDataStream::Qt\_5\_14);

stream<<hash;

if(stream.status() != QDataStream::Ok)

qDebug() << "Ошибка записи";

}

file.close();

}

void MainWindow::readBin(){

QFile file(QCoreApplication::applicationDirPath()+"/positions.bin");

if (file.open(QIODevice::ReadOnly)){

QDataStream stream(&file);

stream.setVersion(QDataStream::Qt\_5\_14);

stream >> hash;

if(stream.status() != QDataStream::Ok)

qDebug() << "Ошибка чтения файла";

}

file.close();

}

void MainWindow::startInTab()

{

if(player->position()==0)

findInHash();

}

void MainWindow::key(){

keyDel = new QShortcut(ui->tab\_1);

keyDel->setKey(Qt::Key\_Delete);

connect(ui->tableView,&QTableView::clicked,[this](const QModelIndex &index){indexRow=index.row();});

connect(keyDel, &QShortcut::activated,this , &MainWindow::indexToDelete);

}

void MainWindow::indexToDelete(){

modelList->removeRows(indexRow,1);

ui->tableView->update();

playlist->removeMedia(indexRow);

ui->lineEdit->setText(modelList->data(modelList->index(playlist->currentIndex(), 0)).toString());

}

void MainWindow::message(){

QMessageBox::information(this,"Информация","Для удаления видео из плейлиста: \n 1.Кликнуть на нужное видео \n 2.Нажать кнопку Del",QMessageBox::Ok);

}

void MainWindow::nextPrevious(){

connect(ui->previous,&QToolButton::clicked,this,&MainWindow::previous);

connect(ui->next,&QToolButton::clicked,this,&MainWindow::next);

}

void MainWindow::previous(){

if(playlist->currentMedia().request().url().toString()!="")

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

playlist->previous();

if(player->currentMedia().isNull())

playlist->previous();

findInHash();

}

void MainWindow::next(){

if(playlist->currentMedia().request().url().toString()!="")

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

playlist->next();

if(player->currentMedia().isNull())

playlist->next();

findInHash();

}

Файл mainwindow.h:

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <QStandardItemModel>

#include <QtMultimedia/QMediaPlayer>

#include <QtMultimedia/QMediaPlaylist>

#include <QtMultimediaWidgets/QVideoWidget>

#include<QFileDialog>

#include<QCoreApplication>

#include<QDebug>

#include<QShortcut>

#include<QMessageBox>

QT\_BEGIN\_NAMESPACE

namespace Ui { class MainWindow; }

QT\_END\_NAMESPACE

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

MainWindow(QWidget \*parent = nullptr);

~MainWindow();

private:

Ui::MainWindow \*ui;

QMediaPlayer \*player;

QVideoWidget \*videoWidget;

QMediaPlaylist \*playlist;

QStandardItemModel \*modelList;

QHash <QUrl,qint64> hash;

QShortcut \*keyDel;

int indexRow;

protected:

/\*\*

\* @brief event Обработка событий

\* @param event Событие

\* @return Событие

\*/

bool event(QEvent \*event) override;

private:

/\*\*

\* @brief openPlaylist Открытие плейлиста

\*/

void openPlaylist();

/\*\*

\* @brief optionsTableView Настройки отоображения плейлиста

\*/

void optionsTableView();

/\*\*

\* @brief optionsVolume Настройки звука

\*/

void optionsVolume();

/\*\*

\* @brief optionsVideoTime Настройки позиции в видео

\*/

void optionsVideoTime();

/\*\*

\* @brief generalOptions Общие настройки внешнего вида

\*/

void generalOptions();

/\*\*

\* @brief generalOptions\_2 Продолжение общих настроек

\*/

void generalOptions\_2();

/\*\*

\* @brief key Горячая клавиша

\*/

void key();

/\*\*

\* @brief nextPrevious Переход к следующему видео

\*/

void nextPrevious();

/\*\*

\* @brief writeBin Запись в бинарный файл

\*/

void writeBin();

/\*\*

\* @brief readBin Чтение бинарного файла

\*/

void readBin();

private slots:

/\*\*

\* @brief playVideo Переход к вкладке с просмотром видео

\*/

void playVideo();

/\*\*

\* @brief pauseVideo Сохранение позиции и остановка видео

\*/

void pauseVideo(int);

/\*\*

\* @brief typeToVolume Отображение значения звука

\*/

void typeToVolume(int);

/\*\*

\* @brief getPosition Установка позиции в проигрывателе

\*/

void getPosition();

/\*\*

\* @brief findInHash Поиск последней точки воспроизведения

\*/

void findInHash();

/\*\*

\* @brief startInTab Просмотр с сохраненной позиции из вкладки с видео

\*/

void startInTab();

/\*\*

\* @brief indexToDelete Удаление видео из плейлиста

\*/

void indexToDelete();

/\*\*

\* @brief message Информационное сообщение

\*/

void message();

/\*\*

\* @brief previous Переход к предыдущему видео

\*/

void previous();

/\*\*

\* @brief next Переход к следующему видео

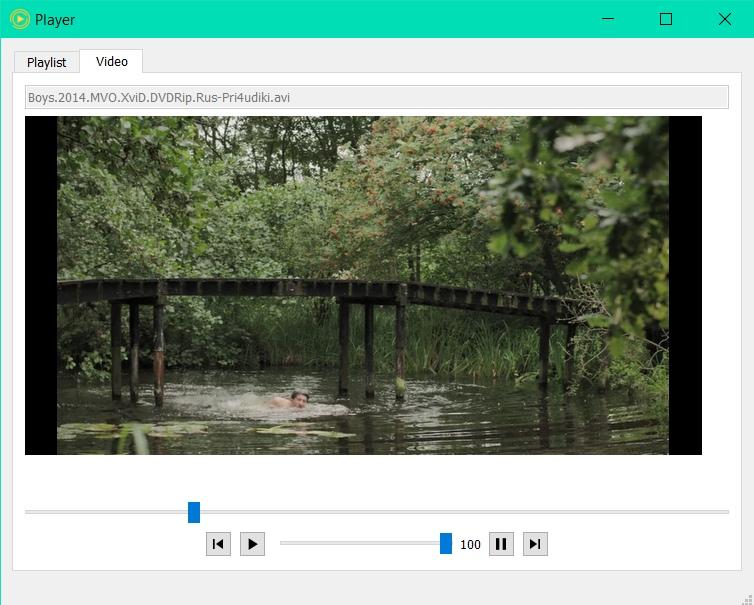
\*/

void next();

};

#endif // MAINWINDOW\_H

В качестве результата работы программы представлены скриншоты выходных файлов на рисунке 3.



*Рисунок 3 – результат работы программы*

# ВЫВОД

В процессе выполнения лабораторной работы были получены навыки работы с мультимедиа в Qt.