课程设计报告

设计题目：33.基于C++的音乐播放器实现

学生姓名：

专 业：智能科学与技术

班 级：智科22-1

学 号：

指导教师：

完成日期：2024-01-08

**合肥工业大学计算机与信息学院**

1. **需求和规格说明**

33：基于C++的音乐播放器实现

**问题描述：**

1.题目要求

(1) 播放器有3个状态：停止状态、播放状态、暂停状态

(2) 播放器有3个按键：播放按键、暂停按键、停止按键

(3) 播放器启动后进入到停止状态，停止状态下只响应播放按键，按下播放按键进入播放态，并开始 播放音乐；播放状态下，不响应播放按键，按下暂停键暂停音乐播放，按下停止键停止音乐播放； 暂停状态下，不响应暂停按键，按下播放键继续播放音乐，按下停止键停止播放音乐。

2.设计目标

此次设计不局限于题目要求，个人的设计目标是实现一个能够至少能够满足个人使用的本地音乐播放器，设想的功能包括

（1）界面与按钮

（2）播放和播放逻辑

（3）歌曲列表

（4）音量与进度条

（5）歌曲界面

（6）记忆功能

**实现要求：**

（1）界面包括歌曲列表界面、歌曲界面，通过按钮来进行界面间的切换。歌曲列表记录目标文件夹中的歌曲，需要能够选中与切换；歌曲界面包括封面和歌词。

（2）按钮包括：歌曲操作与界面操作,歌曲操作包括播放、暂停、上一首、下一首、播放模式、音量、选择文件夹。

（3）实现歌曲通过按钮操作的切换，通过鼠标操作的切换、根据播放模式的自动切换

（4）实现通过文件夹载入播放列表并显示

（5）实现音量调整功能，歌曲进度条显示功能，进度条拖拽功能

（6）实现记忆功能，记录上一次使用时的状态

注：此次项目基于Qt实现

1. **设计**

**系统类图**

（通过窗口对象来演示）



**MainWindow 属性及方法：**

**属性：**

|  |  |  |
| --- | --- | --- |
| 类型 | 名称 | 描述 |
| QIcon | 略(多个) | 保存按钮图标 |
| QMediaPlayer | player | 用于播放和获取媒体信息 |
| QAudioOutout | output | 用于设置音量 |
| bool | play\_status | 用于记录播放状态 |
| QString | current\_name | 当前歌曲名 |
| QPixmap | current\_cover | 当前封面 |
| QList<QUrl> | play\_list | 播放列表 |
| int | play\_list\_mode | #播放模式 |
| float | current\_volume | #音量 |
| QString | current\_file | #当前文件夹 |
| int | play\_list\_index | #当前歌曲序号 |
| qint64 | current\_process | #当前播放时间 |

注：#开头为参数，在程序关闭后会保存

**方法：**

|  |  |  |
| --- | --- | --- |
| 类型 | 名称 | 描述 |
| void | on\_switch\_to\_song\_clicked(); | 转到歌曲界面 |
| void | on\_start\_clicked(); | 播放暂停按钮 |
| void | on\_mode\_clicked(); | 切换播放模式 |
| void | on\_volume\_clicked(); | 音量 |
| void | on\_to\_config\_clicked(); | 转到设置 |
| void | on\_back\_to\_main\_clicked(); | 回到列表 |
| void | on\_volume\_slider\_valueChanged(int value); | 音量调整 |
| void | on\_choose\_file\_clicked(); | 选择文件夹 |
| void | on\_last\_clicked(); | 上一首 |
| void | on\_next\_clicked(); | 下一首 |
| void | on\_listWidget\_itemDoubleClicked(QListWidgetItem \*item); | 选择歌曲 |
| void | update\_song\_status(); | 更新歌曲状态和信息 |
| void | set\_song\_info(); | 设置歌曲信息 |
| void | on\_process\_bar\_valueChanged(int value); | 进度条 |
| void | void on\_list\_clicked(); | 列表 |
| QPixmap | set\_border\_radius(QPixmap cover,int radius); | 圆角封面 |
| void | on\_clear\_clicked(); | 清除列表 |

1. **用户手册**

1.运行程序后，进入主界面（播放列表），点击选择文件夹，选择一个包含.mp3、.wav、.flac的音乐文件夹，载入后，在中间能看到歌曲名。

2.下方为操作区，载入歌曲后激活，包括播放/暂停，切到上下，右侧可调整播放模式，包括顺序、循环、单循环、随机。

4.可通过鼠标选择歌曲并播放。

3.可通过音量按钮可弹出音量条来调整音量，歌曲进度条左右侧显示时间，进度条可拖拽。

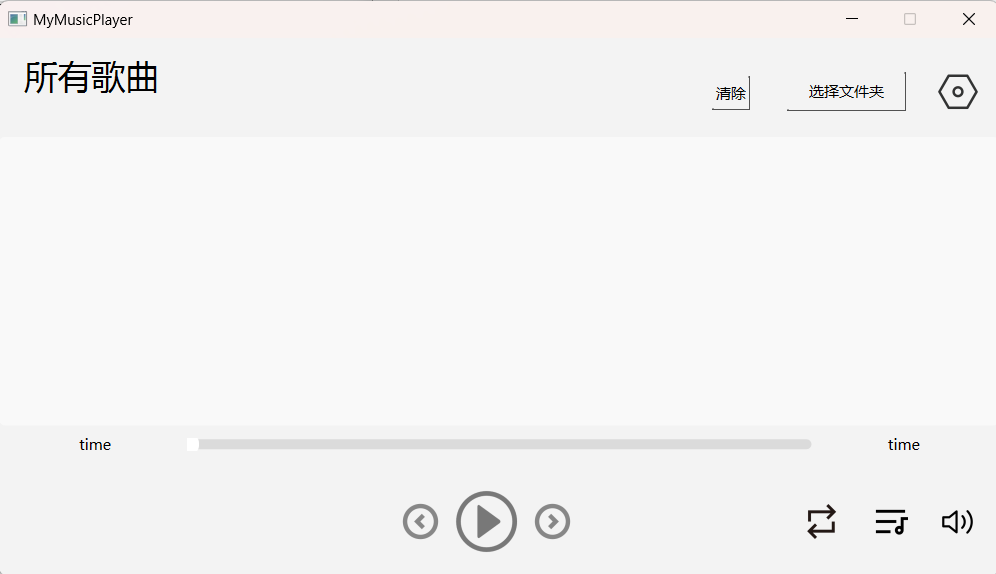
4.左下角显示歌曲封面和曲名，点击加入歌曲界面。

5.关闭程序后再次打开，会继续上次结束时的状态，但播放是默认暂停的。

说明：关于歌曲界面，目前只有封面，由于本地mp3文件不自带歌词，这部分待实现，另外设置按钮点击后加入设置界面，但这一部分由于目前功能还很少，还是空着的。

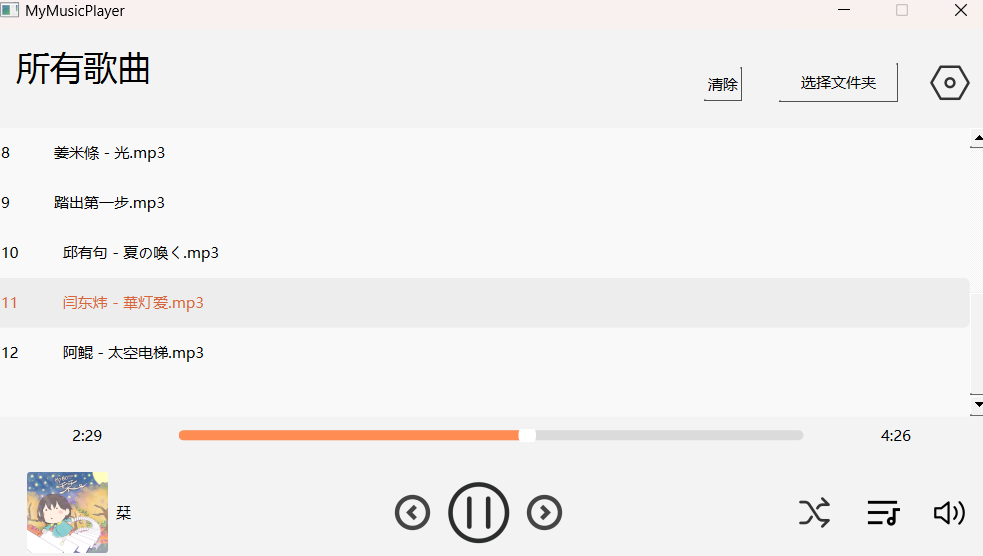
1. **调试及测试**

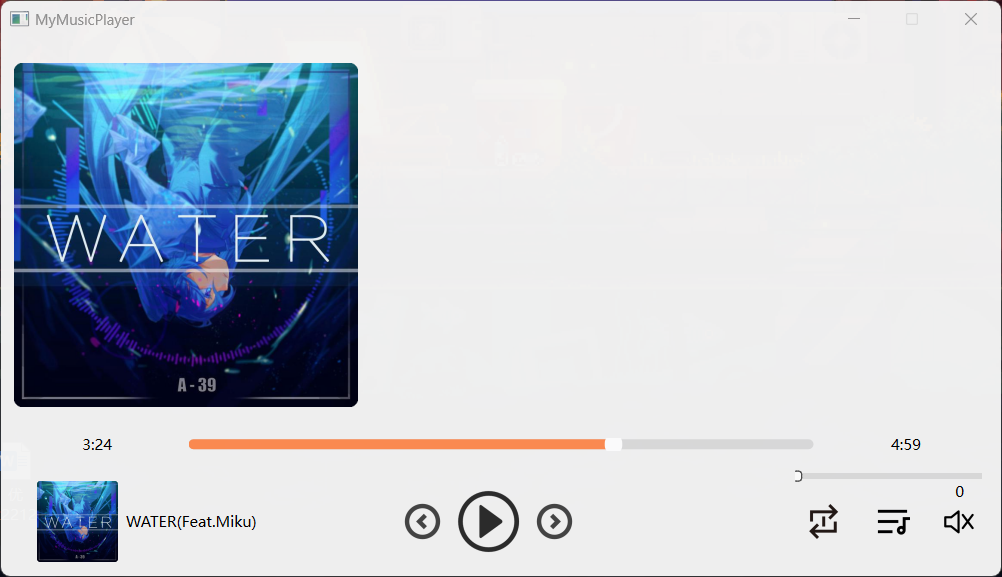
**运行实例：**



图形用户界面, 应用程序

描述已自动生成





**进一步改进**

（1）列表目前只有当前播放列表，后续的想法是设置两个列表，一个显示文件夹的音乐，已经显示在播放的列表。

（2）歌曲界面想加入切换动画和云母效果,有待实现。

（3）设置界面后续可考虑加入选择背景、路径等功能。

（4）制作列表时使用的是ListWidget，这个部件不能显示多列，后续考虑更换以实现显示作者和时长。

**附录一 源程序**

**包括三个部分:mainwindow.h mainwindow.cpp main.cpp**

**mainwindow.h:**

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include<QPushButton>

#include <QtMultimedia/QMediaPlayer>//播放

#include <QtMultimedia/QMediaMetaData>

#include <QtMultimedia/QAudioOutput>

#include <QListWidgetItem>

#include <QUrl> //播放列表

#include <QRandomGenerator>

#include <QImage>

#include <QPainter>

#include <QFrame>

QT\_BEGIN\_NAMESPACE

namespace Ui {

class MainWindow;

}

QT\_END\_NAMESPACE

//主界面类的定义

class MainWindow : public QMainWindow

{

Q\_OBJECT //使用qt元对象

public: //构造和析构函数声明

MainWindow(QWidget \*parent = nullptr);

~MainWindow();

private:

Ui::MainWindow \*ui; //ui指针

/////////////////////////////////////////////////

private slots: //槽函数的声明

void on\_switch\_to\_song\_clicked();

void on\_start\_clicked();

void on\_mode\_clicked();

void on\_volume\_clicked();

void on\_to\_config\_clicked();

void on\_back\_to\_main\_clicked();

void on\_volume\_slider\_valueChanged(int value);

void on\_choose\_file\_clicked();

void on\_last\_clicked();

void on\_next\_clicked();

void on\_listWidget\_itemDoubleClicked(QListWidgetItem \*item);

void update\_song\_status();

void set\_song\_info();

void on\_process\_bar\_valueChanged(int value);

void on\_list\_clicked();

QPixmap set\_border\_radius(QPixmap cover,int radius);

void on\_clear\_clicked();

public: //声明界面中要用到的变量及函数

//////icons

//////////////////////////////////////////////////////

QIcon start\_icon=QIcon(":/assets/play.png");

QIcon pause\_icon=QIcon(":/assets/pause.png");

QIcon last\_icon=QIcon(":/assets/left.png");

QIcon next\_icon=QIcon(":/assets/right.png");

QIcon order\_icon=QIcon(":/assets/order.png");

QIcon loop\_icon=QIcon(":/assets/loop.png");

QIcon single\_icon=QIcon(":/assets/single.png");

QIcon random\_icon=QIcon(":/assets/random.png");

QIcon list\_icon=QIcon(":/assets/list.png");

QIcon volume\_icon=QIcon(":/assets/volume.png");

QIcon mute\_icon=QIcon(":/assets/mute.png");

QIcon config\_icon=QIcon(":/assets/config.png");

QIcon back\_icon=QIcon(":/assets/back.png");

//////////////////////////////////////////////////////

void update\_volume\_icon();

void load\_file(QString filepath);

QMediaPlayer \*player=new QMediaPlayer(this);

QAudioOutput \*output=new QAudioOutput(this);

bool play\_status=false;

QString current\_name; //当前音乐

QPixmap current\_cover;

QList<QUrl> play\_list; //播放列表

//参数

//////////////////////////////////////////////////

int play\_list\_mode=0; //0-order;1-loop;2-single;3-random;

float current\_volume=0.3;

QString current\_file=""; //当前文件夹

int play\_list\_index=0;

qint64 current\_process=0;

// QPixmap background\_pic;

////////////////////////////////////////////////

};

#endif // MAINWINDOW\_H

**mainwindow.cpp:**

#include "mainwindow.h"

#include "./ui\_mainwindow.h"

#include <QFileDialog>//选择文件

#include <QDir>//获取内容

#include <QMediaPlayer>

#include <QSettings>

MainWindow::MainWindow(QWidget \*parent):

QMainWindow(parent),

//主界面的构造函数

ui(new Ui::MainWindow)

{

ui->setupUi(this);

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

this->setWindowTitle("MyMusicPlayer");

this->setMinimumSize(800,430);

this->setMaximumSize(800,430);//限制长宽

ui->stackedWidget->setCurrentIndex(0);

//////////////////////////////

//参数记忆设置

QSettings setting("MyMusicPlayer");

play\_list\_mode=setting.value("play\_list\_mode").toInt();

current\_volume=setting.value("current\_volume").toFloat();

current\_file=setting.value("current\_file").toString();

play\_list\_index=setting.value("play\_list\_index").toInt();

current\_process=setting.value("current\_process",current\_process).toLongLong();

if(!current\_file.isEmpty()){

load\_file(current\_file);

}

/////////////////////////////

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

//播放设置

ui->switch\_to\_song->setDisabled(true);

ui->start->setDisabled(true);

ui->last->setDisabled(true);

ui->next->setDisabled(true);

ui->process\_bar->setDisabled(true);

player->setAudioOutput(output);

output->setVolume(current\_volume);

connect(MainWindow::player, SIGNAL(mediaStatusChanged(QMediaPlayer::MediaStatus)), this, SLOT(set\_song\_info()));

//// 音量条

ui->volume\_slider->setParent(this);

ui->volume\_slider->setRange(0,100);

ui->volume\_slider->setValue(current\_volume\*100.0);

ui->volume\_slider->setGeometry(635,345,150,15);

ui->volume\_slider->hide();

ui->volume\_dis->setAlignment(Qt::AlignCenter);

ui->volume\_dis->hide();

////////////////////////////////////////////////////////////////////////////////

QString button\_style="QPushButton { background-color: transparent; border: none; }"

"QPushButton:hover {background-color: rgba(0, 0, 0, 0.1);}"

"QPushButton:pressed { background-color: rgba(0, 0, 0, 0.05); }";

////进度条

connect(player,&QMediaPlayer::durationChanged,this,[=](qint64 duration){

ui->total\_time->setText(QString("%1:%2").arg( QString::number(duration/60000),QString::number(duration%60000/1000).rightJustified(2,'0')) );

ui->process\_bar->setRange(0,duration);

});

connect(player,&QMediaPlayer::positionChanged,this,[=](qint64 pos){

static int t=0;

if (t>10)

{

ui->current\_time->setText(QString("%1:%2").arg( QString::number(pos/60000),QString::number(pos%60000/1000).rightJustified(2,'0')) );

ui->process\_bar->setValue(pos);

t=0;

}

// qInfo()<<t;

t++;

});

ui->current\_time->setAlignment(Qt::AlignCenter);

ui->total\_time->setAlignment(Qt::AlignCenter);

////to song\_face button

ui->song\_face->setScaledContents(true);

ui->music\_pic->setScaledContents(true);

ui->music\_name->setAlignment(Qt::AlignLeft|Qt::AlignVCenter);

ui->switch\_to\_song->setParent(this);

ui->switch\_to\_song->setGeometry(10,349,210,75);

ui->switch\_to\_song->raise();

ui->switch\_to\_song->setStyleSheet(button\_style);

///Button\_icon

//////////////////////////////////////////////////////////////////////////////////

{

if(play\_status){ui->start->setIcon(pause\_icon);}

else{ui->start->setIcon(start\_icon);}

ui->start->setIconSize(QSize(52,52));

ui->start->setStyleSheet(button\_style);

ui->last->setIcon(last\_icon);

ui->last->setIconSize(QSize(42,42));

ui->last->setStyleSheet(button\_style);

ui->next->setIcon(next\_icon);

ui->next->setIconSize(QSize(42,42));

ui->next->setStyleSheet(button\_style);

switch(play\_list\_mode){

case 0:

ui->mode->setIcon(order\_icon);

break;

case 1:

ui->mode->setIcon(loop\_icon);

break;

case 2:

ui->mode->setIcon(single\_icon);

break;

case 3:

ui->mode->setIcon(random\_icon);

break;

}

ui->mode->setIconSize(QSize(28,28));

ui->mode->setStyleSheet(button\_style);

ui->list->setIcon(list\_icon);

ui->list->setIconSize(QSize(28,28));

ui->list->setStyleSheet(button\_style);

if(current\_volume){ui->volume->setIcon(volume\_icon);}

else{ui->volume->setIcon(mute\_icon);}

ui->volume->setIconSize(QSize(28,28));

ui->volume->setStyleSheet(button\_style);

ui->to\_config->setIcon(config\_icon);

ui->to\_config->setIconSize(QSize(32,32));

ui->to\_config->setStyleSheet(button\_style);

ui->back\_to\_main->setIcon(back\_icon);

ui->back\_to\_main->setIconSize(QSize(32,32));

ui->back\_to\_main->setStyleSheet(button\_style);

}

//////////////////////////////////////////////////////////////////////////////////

}

MainWindow::~MainWindow(){

delete ui;

QSettings setting("MyMusicPlayer");

setting.setValue("play\_list\_mode",play\_list\_mode);

setting.setValue("current\_volume",current\_volume);

setting.setValue("current\_file",current\_file);

setting.setValue("play\_list\_index",play\_list\_index);

current\_process=player->position();

setting.setValue("current\_process",current\_process);

}

///////////////////////////////////////////////////////////////////////////////////////////////////

///////////////////////////////////////////////////////////////////////////////////////////////////

///////////////////////////////////////////////////////////////////////////////////////////////////

//歌曲界面

void MainWindow::on\_switch\_to\_song\_clicked(){

if(ui->stackedWidget->currentIndex()!=1){

ui->stackedWidget->setCurrentIndex(1);

setWindowOpacity(0.98);

}

else{

ui->stackedWidget->setCurrentIndex(0);

setWindowOpacity(1.0);

}

}

//播放按钮

void MainWindow::on\_start\_clicked()

{

if(play\_status){ //播放时

ui->start->setIcon(start\_icon);

play\_status=false;

player->pause();

}

else{ //暂停时

ui->start->setIcon(pause\_icon);

play\_status=true;

player->play();

}

}

//播放模式

void MainWindow::on\_mode\_clicked()

{

play\_list\_mode+=1;

if(play\_list\_mode>3){play\_list\_mode=0;}

switch(play\_list\_mode){

case 0:

ui->mode->setIcon(order\_icon);

break;

case 1:

ui->mode->setIcon(loop\_icon);

break;

case 2:

ui->mode->setIcon(single\_icon);

break;

case 3:

ui->mode->setIcon(random\_icon);

break;

}

}

void MainWindow::on\_list\_clicked(){

ui->stackedWidget->setCurrentIndex(0);

setWindowOpacity(1.0);

}

//进度条

void MainWindow::on\_process\_bar\_valueChanged(int value){

if(player->duration()-value>300){ //防止拖到尾部连续切歌

player->setPosition(value);}

else{

player->setPosition(player->duration()-300);

}

}

//设置

void MainWindow::on\_to\_config\_clicked(){

ui->stackedWidget->setCurrentIndex(2);

}

void MainWindow::on\_back\_to\_main\_clicked(){

ui->stackedWidget->setCurrentIndex(0);

}

//音量调整

////////////////////////////////////////////////////////////////////////////////

void MainWindow::on\_volume\_clicked()

{

if(ui->volume\_slider->isHidden()){

ui->volume\_slider->show();

ui->volume\_dis->setText(QString::number(current\_volume\*100.0));

ui->volume\_dis->show();

}

else{

ui->volume\_slider->hide();

ui->volume\_dis->hide();

}

}

void MainWindow::update\_volume\_icon(){

if(current\_volume){

ui->volume->setIcon(volume\_icon);

}

else{

ui->volume->setIcon(mute\_icon);

}

}

void MainWindow::on\_volume\_slider\_valueChanged(int value)

{

current\_volume=value/100.0;

output->setVolume(current\_volume);

ui->volume\_dis->setText(QString::number(current\_volume\*100.0));

update\_volume\_icon();

}

////////////////////////////////////////////////////////////////////////////////

//歌曲信息

//////////////////////////////////////////////////////////////////////////////////////////

////圆角图

QPixmap MainWindow::set\_border\_radius(QPixmap cover,int radius){

QBitmap mask(cover.size());

mask.fill(Qt::color0);

QPainter painter(&mask);

painter.setRenderHint(QPainter::Antialiasing);

painter.setBrush(Qt::color1);

painter.drawRoundedRect(mask.rect(), radius, radius);

cover.setMask(mask);

return cover;

}

////

////歌曲更新

void MainWindow::set\_song\_info(){

if(player->isAvailable()){

static bool when\_launch=true;

if(current\_process!=0){

if(when\_launch){

player->setPosition(current\_process);

ui->process\_bar->setValue(current\_process);

when\_launch=false;

}

}

////封面和标题

if(player->mediaStatus()==QMediaPlayer::LoadedMedia){

QMediaMetaData mediaData = player->metaData();

current\_name = mediaData.value(QMediaMetaData::Title).toString();

if(!current\_name.isEmpty()){

ui->music\_name->setText(current\_name);

}

else{

ui->music\_name->setText(player->source().fileName());

}

QVariant coverdata1 = mediaData.value(QMediaMetaData::CoverArtImage);

QVariant coverdata2 = mediaData.value(QMediaMetaData::ThumbnailImage);

ui->song\_face->show();

if(!coverdata1.isNull()){

current\_cover = qvariant\_cast<QPixmap>(coverdata1);

ui->music\_pic->setPixmap(set\_border\_radius(current\_cover,30));

ui->song\_face->setPixmap(set\_border\_radius(current\_cover,20));

}

else if(!coverdata2.isNull()){

current\_cover = qvariant\_cast<QPixmap>(coverdata2);

ui->music\_pic->setPixmap(set\_border\_radius(current\_cover,30));

ui->song\_face->setPixmap(set\_border\_radius(current\_cover,20));

}

else{

ui->music\_pic->setStyleSheet("background:#FFFFFF;");

ui->music\_pic->setText("No Image");

ui->song\_face->hide();

}

ui->switch\_to\_song->setDisabled(false);

ui->start->setDisabled(false);

ui->last->setDisabled(false);

ui->next->setDisabled(false);

ui->process\_bar->setDisabled(false);

}

////切歌

if(player->mediaStatus()==QMediaPlayer::EndOfMedia){

if(play\_status){

if(play\_list\_mode==0){ //order

if(play\_list\_index<play\_list.length()-1){

play\_list\_index+=1;

update\_song\_status();

}

else{

player->pause();

ui->start->setIcon(start\_icon);

}

}

else if(play\_list\_mode==2){//single

update\_song\_status();

}

else{

on\_next\_clicked();

}

}

}

}

}

//////////////////////////////////////////////////////////////////////////////////////////

//选择文件

void MainWindow::load\_file(QString filepath){

current\_file=filepath; //更新路径

QDir dir(filepath); //获取歌曲信息

auto music\_list=dir.entryList(QStringList()<<"\*.mp3"<<"\*.wav"<<"\*.flac");

//播放列表操作

//1.将歌曲添加入播放列表

int num=1;

for(auto &music:music\_list){

ui->listWidget->addItem(QString::number(num)+" "+music);

play\_list.append(QUrl::fromLocalFile(filepath+"/"+music));

num++;

}

//2.载入当前歌曲

if(play\_list\_index>play\_list.length()-1 ||play\_list\_index<0){play\_list\_index=0;}//防止意外修改

ui->listWidget->setCurrentRow(play\_list\_index);

play\_list\_index=ui->listWidget->currentRow();

//3.设置播放源

player->setSource(play\_list[play\_list\_index]);

if(play\_status){

player->play();}

}

void MainWindow::on\_choose\_file\_clicked(){

QString filepath=QFileDialog::getExistingDirectory(this,QString::fromLocal8Bit("选择音乐文件夹"),

QCoreApplication::applicationFilePath());

if(!filepath.isEmpty()){

current\_file=filepath;

load\_file(filepath);

}

}

//手动歌曲切换

//////////////////////////////////////////////////////////////////////////////////////////

void MainWindow::update\_song\_status(){

ui->listWidget->setCurrentRow(play\_list\_index);

player->setSource(play\_list[play\_list\_index]);

if(play\_status){player->play();}

}

void MainWindow::on\_last\_clicked(){ //0-order;1-loop;2-single;3-random;

switch(play\_list\_mode){

case 0:

if(play\_list\_index>0){

play\_list\_index-=1;

}

break;

case 1:

if(play\_list\_index>0){

play\_list\_index-=1;

}

else{

play\_list\_index=play\_list.length()-1;

}

break;

case 2:

if(play\_list\_index>0){

play\_list\_index-=1;

}

else{

play\_list\_index=play\_list.length()-1;

}

break;

case 3:

QRandomGenerator \*ge = QRandomGenerator::global();

play\_list\_index=ge->bounded(0,play\_list.length());

break;

}

update\_song\_status();

}

void MainWindow::on\_next\_clicked(){

switch(play\_list\_mode){

case 0:

if(play\_list\_index<play\_list.length()-1){

play\_list\_index+=1;

}

break;

case 1:

if(play\_list\_index<play\_list.length()-1){

play\_list\_index+=1;

}

else{

play\_list\_index=0;

}

break;

case 2:

if(play\_list\_index<play\_list.length()-1){

play\_list\_index+=1;

}

else{

play\_list\_index=0;

}

break;

case 3:

QRandomGenerator \*ge = QRandomGenerator::global();

play\_list\_index=ge->bounded(0,play\_list.length());

break;

}

update\_song\_status();

}

void MainWindow::on\_listWidget\_itemDoubleClicked(QListWidgetItem \*item)

{

play\_list\_index=ui->listWidget->row(item);

play\_status=true;

ui->start->setIcon(pause\_icon);

update\_song\_status();

}

//////////////////////////////////////////////////////////////////////////////////////////

void MainWindow::on\_clear\_clicked()

{

player->setSource(QUrl());

current\_file="";

play\_list.clear();

play\_list\_index=0;

ui->listWidget->clear();

set\_song\_info();

ui->start->setDisabled(true);

ui->last->setDisabled(true);

ui->next->setDisabled(true);

ui->music\_name->clear();

ui->music\_pic->clear();

ui->switch\_to\_song->setDisabled(true);

}

**main.cpp**

#include "mainwindow.h"

#include <QApplication>

#include<QTextStream>

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

QApplication a(argc, argv);

MainWindow w;

w.show();

return a.exec();

}

**附录二 使用到的icon**

