**摘 要**

现在的我们正处于一个娱乐多元化的时代，越来越多的人们开始注重自己的健康与生活娱乐方式，然而，音乐是每个人必不可少的娱乐项目，随着市面上音乐播放器的大量涌现，人们对其要求也随之增高。因此，选择设计音乐播放器系统，是希望在学习多种播放器基本功能的基础上，对其进行优化和创新。

本文从利用的工具，能够实现的功能以及整体架构和遇到的问题与解决的办法等方面，对该音乐播放器系统进行详细介绍。

该系统所用的开发工具是QT，开发的语言是C++，利用C++面向对象的编程方式，设计良好的用户图形界面，带给用户更好的使用体验。本系统的功能主要分为三个方面，一是音乐播放器的基本功能，其中包括音乐的播放、暂停、上一首、下一首、快进、快退等；二是音乐播放器的控制功能，其中包括音量的控制、歌词的控制、歌曲进度的控制、皮肤的切换；三是音乐播放器的网络下载功能，其中包括歌曲的下载、歌词的下载。该系统采用QT作为平台，通过Mysql数据库进行数据存储，利用多平台移植性，使系统可以在多个平台上的运用，从而提高软件的通用性，同时利用多元化界面之间的切换，不但增加了软件趣味性，而且提高了用户体验感。

**关键词：**QT；C++；Mysql数据库；音乐播放器

、

**Abstract**

Nowadays,we are in an era of entertainment diverse,more and more people begin to pay attention to their health and the life of entertainment,howerver,music is an indispensable entertainment project for everyone,with the emergence of a large number of music players on the market,the people requirement of performance better and better,so,choose a system designed for nusic player,I hope to further optimize and innovate that base on the study of the basic functions of a variety of players.

The project introduce the system from the use of tools,can achieve the functions as well as the overall architecture and the problems and solution,etc.,until the design has carried on the detailed instructions.

The system used by the development tool is QT,the development of the language is C++,the use of C++ object-oriented programming,design a good user interface,giving users a better experience.The main function of this system is mainly divided into three aspects,one is the basic functions of the music player,includeing music playback,pause a song,the next song,fast forward,rewind and other functions;two is to control the music player,including volume control,lyric control,control the progress of the song skin,switching;three is the function of network to download music player,including song download,lyrics download.The system uses QT as the platform,Through the Mysql database for data storage，using its multiplatform portability,realize the use on multiple platforms,improve the versatility,by switching the interface between diversification,increased interest,improve the user experience.

**Key words**：QT; C++; Mysql Database;Music player

目录

[1 引言 5](#_Toc22395)

[1.1课题背景 6](#_Toc28358)

[1.2 课题目的与意义 6](#_Toc14463)

[1.3 国内外发展现状 6](#_Toc16629)

[1.4 开发工具和技术 7](#_Toc31074)

[2 软件的需求分析 8](#_Toc4770)

[2.1 功能需求分析 8](#_Toc6010)

[2.1.1 播放器的基本需求 8](#_Toc3740)

[2.1.2 播放器的播放控制需求 8](#_Toc29158)

[2.1.3 播放器的列表管理需求 8](#_Toc8425)

[2.1.4 播放器的界面友好需求 8](#_Toc12400)

[2.2 系统需求分析 9](#_Toc9867)

[2.2.1 系统界面需求 9](#_Toc11612)

[2.2.2 系统性能需求 9](#_Toc15149)

[2.2.3 运行环境要求 9](#_Toc14428)

[3 基于Qt的音乐播放器项目介绍 10](#_Toc25752)

[3.1 什么是Qt？ 10](#_Toc28023)

[3.1.1 Qt简介 10](#_Toc32593)

[3.1.2 Qt特性 10](#_Toc1235)

[3.2 搭建Qt开发环境 10](#_Toc16844)

[3.3 Qt音乐播放器的工程 10](#_Toc26193)

[4 音乐播放器总体设计 12](#_Toc23203)

[4.1 音乐播放器界面模块设计 13](#_Toc15224)

[4.2 音乐播放器功能模块设计 13](#_Toc30029)

[4.3 网络传输数据库模块设计 14](#_Toc12741)

[4.4 音乐播放器总体设计 14](#_Toc424)

[5 音乐播放器功能详细设计 15](#_Toc30861)

[5.1 音乐播放器主界面功能实现 16](#_Toc15585)

[5.1.1 播放器主界面 16](#_Toc7887)

[5.1.2 播放器界面进度条音轨的实现 17](#_Toc11893)

[5.1.3 播放器播放上下曲切换，暂停，停止等功能的实现 18](#_Toc22693)

[5.2 播放列表功能 19](#_Toc8800)

[5.2.1 播放器列表的添加功能 19](#_Toc12600)

[5.2.2 播放器列表的删除功能 21](#_Toc12737)

[5.2.3 播放器列表的搜索功能 22](#_Toc5478)

[5.2.4 播放器界面皮肤更换功能 23](#_Toc13674)

[5.3 菜单功能 24](#_Toc17526)

[5.3.1 菜单界面 24](#_Toc6718)

[5.3.2 菜单功能实现 25](#_Toc24462)

[5.4 播放设置界面 27](#_Toc3413)

[5.4.1 歌词显示实现 27](#_Toc21966)

[5.5 网络下载功能 30](#_Toc26154)

[5.5.1 连接服务器 30](#_Toc6708)

[5.5.2 下载功能的实现 32](#_Toc10089)

[5.6 数据存储方式 34](#_Toc14779)

[5.6.1 MySQL数据库存储 34](#_Toc29472)

[6 系统测试与性能分析 35](#_Toc23377)

[6.1 系统测试 36](#_Toc30032)

[6.2 性能分析 44](#_Toc16531)

[结论 45](#_Toc8245)

[参考文献 45](#_Toc21886)

[致谢 47](#_Toc12232)

1 引言

## 1.1课题背景

如今音乐播放器的应用十分广泛，大到公司企业的商业开发用途，小到每家每户的生活娱乐方式，人们每天都离不开音乐的相伴，而音乐播放器更是人们所选择的最为方便的娱乐工具。[[1]](#endnote-0)

目前市面上的音乐播放器种类繁多，但大多数都已商业化。例如，在下载某公司的音乐播放器产品时，播放器常常会捆绑多种应用软件，这就造成了用户的反感。而如果是一些开源的音乐播放软件，又会出现一些恶意链接，使用户所点击的并非是自己需要的软件，提高误操作率。

因此，无恶意插件及广告的绿色音乐播放器系统，能使用户更好的享受音乐带来的欢乐，同时，简单、易操作的界面也让用户能够更快的熟悉与运用。

## 1.2 课题目的与意义

本课题的目的是设计并开发一款简单、易操作，但又不失功能性的音乐播放器。主要实现的功能有：播放多种格式的音频，以及歌曲控制控制，其中包括：播放、暂停、上一首、下一首、音量和背景的切换，可以选择播放模式，以及运行的进度条、歌词等。

一款简单易操作的音乐播放器无疑是大家所期待的，会对今后的娱乐生活带来划时代的意义。

## 1.3 国内外发展现状

自90年代初MP3音频压缩技术诞生以来，人们的生活便离不开伴随的音乐。随着数字音乐的发展，好多音乐方向也陷入了瓶颈，然而音乐播放器更是唯一能够达到数字音乐效果的工具。因此，开发一款音乐播放器有着极其重要的意义。

根据全球的市场环境分析来看，目前数字音乐产业呈现快速的增长势头，而整个数字音乐的销售额要比整个唱片业的销售额更高，可以看出，数字音乐的发展是必然趋势，如今，更有英美等国家正采用强有力的政策，支持数字音乐的发展。

根据全球的技术环境分析来看，大数据时代的来临无疑为数字音乐新添一股力量，现在，用户可以把收藏的音乐存储到云端，这样不仅方便用户随时随地可以享受音乐，同时，更不限制播放音乐的平台，更好的实现了音乐的共享。

对于美国的苹果来说，大家所了解的苹果的应用均是由公司自带的应用商店来进行下载的，而苹果自带的音乐播放器都是采取云端存储的，用户可以通过云端进行海量的搜索，并对喜欢的音乐反复播放，方便了用户不同内存的尴尬问题。

面对国内的播放器来说，我最为喜欢的便是酷我音乐盒，它通过简单，炫酷的操作界面第一眼吸引了广大用户，酷我也同样采用云存储，方便用户的PC端和移动端的数据共享，绝大程度的方便了用户的使用需求。

随着人们生活方式的提高，国内外对娱乐产业更是尤为重视，而作为最通用的音乐来说，设计开发音乐播放工具尤为重要，也同时让这个工具应用更加广泛，更能成为自由音乐者的摇篮。

## 1.4 开发工具和技术

本课题使用的开发平台是QT，QT是诺基亚旗下的多功能开发框架，它的移植性高，功能强大，也易于扩展。同时，它所拥有的库十分广泛，为开发者提供了强有力的条件，对于开发者来说，QT是开发快捷，短小而精悍的应用的最好选择。

实现这次课题的主要开发语言为C++，C++作为一种面向对象的语言被大家广泛运用，高模块化，封装性好是运用C++进行编程的优势，通过面向用户的方式方法，使开发的软件更加贴合用户的需要，同时也能更好的提高软件的重用性和扩展性。

# 

# 2 软件的需求分析

## 2.1 功能需求分析

### 2.1.1 播放器的基本需求

对于此音乐播放器的基本需求，主要从三大方面来考虑功能界面的实现，有操作界面的初始化、显示选择功能界面以及操作功能界面，通过这三个功能界面的实现，使播放器的整体功能显示更加流畅，各个功能衔接更加畅快。

### 2.1.2 播放器的播放控制需求

随着音乐播放器的大量涌现，音乐播放器的基本控制需求尤为重要，主要通过以下基本功能实现。其中有歌曲的播放与暂停、歌曲进度的快进与快退、歌曲之间的上下首切换、歌名的滚动显示、音量调节大小和静音模式之间的切换、以及播放模式的调节。通过实现基本控制需求，使播放器完成所需基本功能。

### 2.1.3 播放器的列表管理需求

此音乐播放器采用两个列表进行歌曲存储，分为本地歌曲列表和网络歌曲列表。歌曲导入列表有两种方式进行导入，一种为直接使用拖拽的方式进行导入，另一种是点击添加文件夹进行导入。对于导入到列表中的歌曲可以拥有双击播放、删除当前歌曲以及清空列表的功能。采用两个列表的方式存储歌曲，可使歌曲分类更加明确，提升用户体验感。

### 2.1.4 播放器的界面友好需求

### 作为客户端，界面需求是用户十分在意的，所以此音乐播放器会从主界面、皮肤更改界面、歌词列表界面、本地歌曲列表界面以及网络列表界面这五个界面需求着手考虑实现相关功能，来提升用户的操作感。

## 2.2 系统需求分析

### 2.2.1 系统界面需求

对于此系统的界面需求有以下三点：

（1）主观界面要美观，皮肤背景简单大方；

（2）按钮设计要合理，用户的可操作感要好；

（3）列表显示要清晰，明确显示需要字段。

用户界面一般都是为了提供系统与用户之间相互交互的接口，同样，只有接口恰当、界面更加良好，才会给用户一种很舒适的体验感。

### 2.2.2 系统性能需求

#### 2.2.2.1 系统响应时间需求

对于本系统的性能要求有两点：

（1）在音乐播放的时候，要求系统响应时间不超过2秒；

（2）在音乐暂停的时候，要求系统响应时间不超过1秒。

#### 2.2.2.2 系统流畅性需求

对于系统的流畅性，在播放本地列表中的歌曲会比较流畅。

#### 2.2.2.3 系统可靠性需求

本系统采用多模块化的设计方法，程序具有较强的移植性和扩展性，提供了接口添加新功能，有利于系统未来的扩展，使系统可在多平台上运行，在不影响其他软件同时使用的情况下，不占用多余内存，增加系统可靠性。

### 2.2.3 运行环境要求

本系统的运行环境为Windows7的操作系统，运行在QT5.4的平台上，使用的数据库为Mysql数据库。

# 3 基于Qt的音乐播放器项目介绍

## 3.1 什么是Qt？

### 3.1.1 Qt简介

QT最先是由诺基亚公司利用C++开发的一种可视化开发平台，作为一种纯面向对象的开发软件，从1996年起，QT正式进入了商业领域，同时它的跨平台的图形界面应用框架也多为程序开发者使用，利用QT，开发者可以更好地实现用户所需要的产品需求，通过良好的用户交互界面，使用户的体验感不断提高，为软件市场创造了很大的价值。

### 3.1.2 Qt特性

QT是一种跨平台、有较高移植特性的可视化开发软件，而且，相比其他软件来说，QT的程序更加短小精悍，功能更加完善，通过调用QT中不同的类，可以实现用户所需要的多种功能，而QT的跨平台更为用户解决了很大难题，QT的程序可以在多个平台上使用，真正达到一份代码，多次利用、多次实现的效果。

## 3.2 搭建Qt开发环境

本次开发利用的QT环境是QT 5.4环境，对于QT 5.4来说，增加了很多更为强大的功能类，可以更好的实现用户的需求，比如QT可以跟其他的框架进行交互，添加了QOpenGLWindow类，很方便的通过OpenGL将内容绘制到QWindow，也有加入了原生文件对话框的支持等多种可用类。

## 3.3 Qt音乐播放器的工程

QT作为一个集成软件开发的平台，引入了信号与槽的机制，在编译过程中只能基于工程进行编译，所以创建工程主要分为三点。首先，选择QT部件应用程序创建项目；其次，填写项目名称以及选择存储路径，方便项目的管理；最后，编写代码完成创建。

具体步骤显示如下图1、图2、图3。

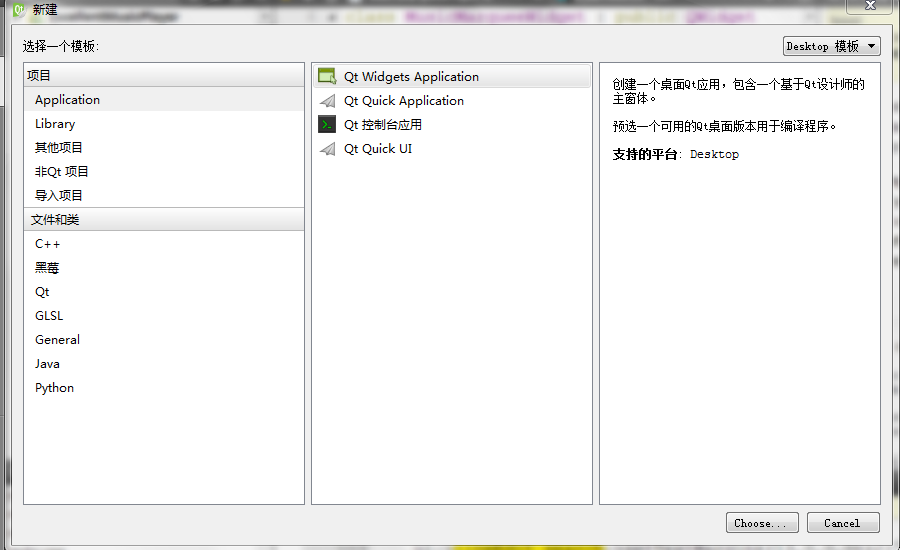


图1 新建一个工程

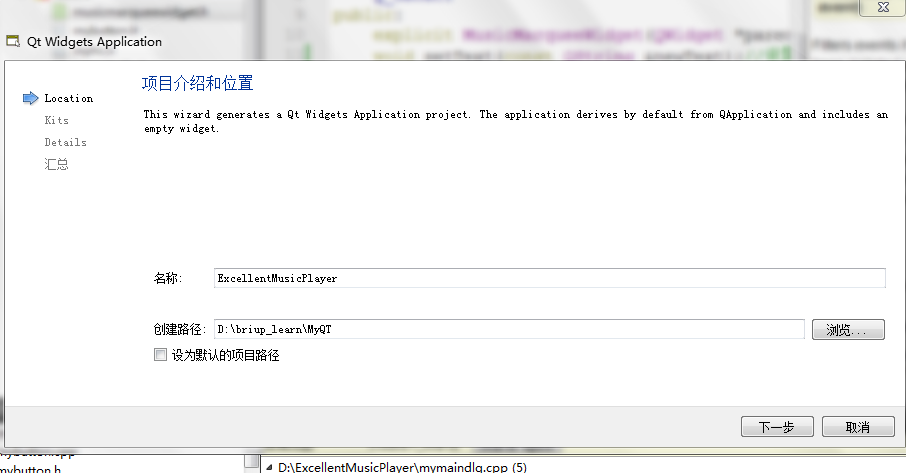


图2 创建项目名称和保存路径

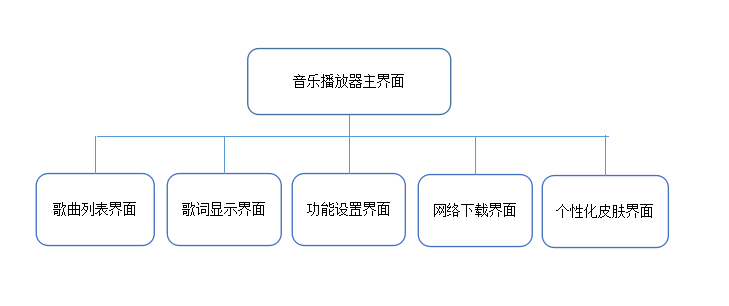


图3 完成创建

# 4 音乐播放器总体设计

## 4.1 音乐播放器界面模块设计

音乐播放器界面模块主要包括歌曲列表界面模块、歌词显示界面模块、功能设置界面模块、网络下载界面模块以及个性化皮肤界面模块。界面模块设计流程见下图4。

 图4 界面模块设计

## 4.2 音乐播放器功能模块设计

音乐播放器功能模块设计分为六大类，分别为皮肤切换功能、音量控制功能、歌曲进度控制功能、歌词进度显示功能、播放模式控制功能、歌曲切换功能、网络下载功能以及歌曲列表控制功能。通过六大功能共同作用，使该播放器的功能实现更加完善。具体功能模块如下图5。

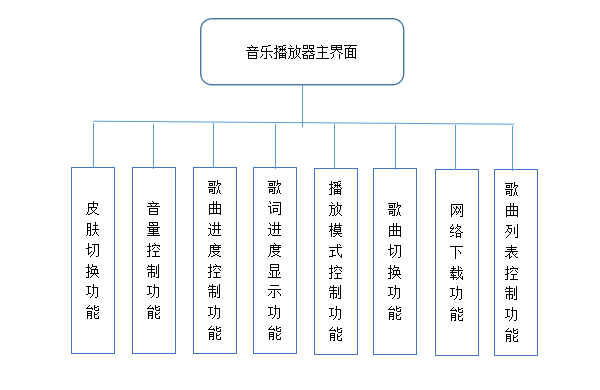


图5 功能模块设计

## 4.3 网络传输数据库模块设计

该音乐播放器的网络传输模块是以TCP协议为主，传输采用Socket进行数据传输，具体流程如下图6所示。

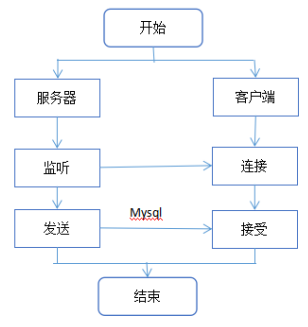


图6 网络传输数据库设计

## 4.4 音乐播放器总体设计

音乐播放器总体设计主要从四大方面着手考虑，分为歌曲列表模块、歌曲播放模式模块、全局设置模块以及歌词显示模块，通过这四大模块的实现，贯穿整体设计，使设计逻辑更加清晰，实现功能更加顺畅。具体流程如下图7。

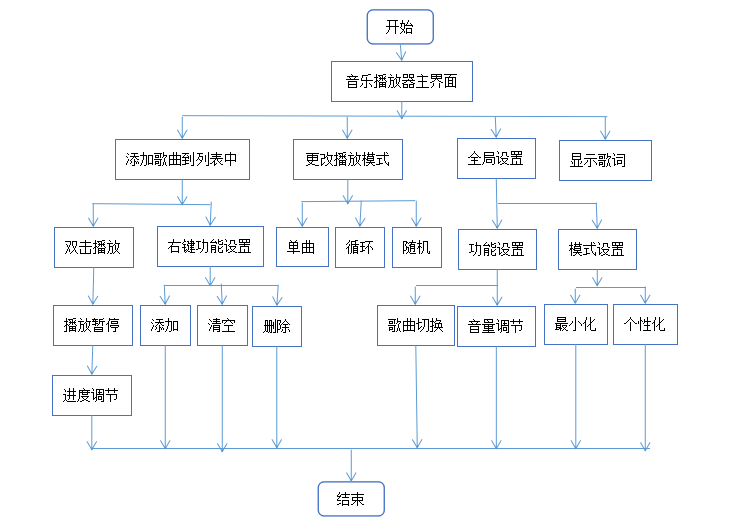


图7 总体设计流程图

# 5 音乐播放器功能详细设计

## 5.1 音乐播放器主界面功能实现

### 5.1.1 播放器主界面

播放器的主界面是由播放器客户端和下载客户端组成，风格大气简单，易于操作，功能齐全。

界面按钮通过PhotoShop软件制作为背景透明，让界面看起来更加美观和谐，整体效果先通过水平布局方式使其整体摆放，最后通过网格布局对整体格局布局，使其放大缩小都随之共同变化。

主要实现代码如下：

this->resize(250,580);

this->setWindowOpacity(0.7);//全体透明

this->setAttribute(Qt::WA\_TranslucentBackground,true);

mytabw = new QTableView(this);

mytabw->move(20,40);

mytabw->resize(210,520);



图8 播放器主界面

### 5.1.2 播放器界面进度条音轨的实现

进度条音轨的实现主要通过重写时间事件timerEvent(QTimerEvent \*event)，同时重写鼠标点击事件函数mousePressEvent(QMouseEvent \*event)，对音轨的定位更有效的把握，使其最终可以通过鼠标点击定位歌曲进度，控制歌曲的快进和快退。

主要代码实现如下：

ui->horizontalSlider\_vol->installEventFilter(this);//用于点击slider即到（声音）

ui->horizontalSlider->installEventFilter(this);//用于点击slider即到（歌曲进度）

connect(ui->horizontalSlider, SIGNAL(sliderMoved(int)), this, SLOT(setPosition(int)));

connect(player, SIGNAL(positionChanged(qint64)), this, SLOT(positionChanged(qint64)));

connect(player, SIGNAL(durationChanged(qint64)), this, SLOT(durationChanged(qint64)));

void MyMainDlg::durationChanged(qint64 duration)//作全局变量保存总时间长度然后加到lable

{

ui->horizontalSlider->setRange(0, duration);

QTime total\_time(0, (duration/60000)%60, (duration/1000)%60);

QString format = "mm:ss";

total\_time.toString(format);

totaldisplay=total\_time.toString(format);

}

void MyMainDlg::setPosition(int position)//slider的位置改变就设置

{

player->setPosition(position);

}



图9 进度条音轨

### 5.1.3 播放器播放上一首歌曲切换，暂停，停止等功能的实现

播放器点击播放按钮播放当前歌曲，此时播放按钮变为暂停按钮图案，为暂停歌曲做准备，同时上下曲随时切换。

主要实现代码如下：

void MyMainDlg::on\_pushButton\_backward\_clicked()

{

if(player->isAudioAvailable()==true)//没有音乐输入就没法按按钮

{

//改变playpause图片

if(nowplaylist==1)

{

mymaintable->mysetprevious();

player->setMedia(mymaintable->myCurrentMedia());

}

else

{

mynetmaintable->mysetprevious();

player->setMedia(mynetmaintable->myCurrentMedia());

}

player->play();

ui->pushButton\_playpause->setToolTip("暂停");

}

}

void MyMainDlg::on\_pushButton\_playpause\_clicked()

{

//设置play和pause的按钮图片状态

if(player->state()==QMediaPlayer::PlayingState)

{

player->pause();

}

else

{

if(player->isAudioAvailable()==true)//没有音乐输入就没法按按钮

{

player->play();

}

}

}



图10 上下首切换以及播放与暂停

## 5.2 播放列表功能

### 5.2.1 播放列表添加功能

当前本地播放列表添加歌曲，有以下两种方式：

（1）添加文件以及文件夹

通过访问文件夹的方式，添加其中歌曲与歌词文件。

（2）拖拽文件直接进列表中

点击当前选择的歌曲文件，直接通过拖拽方式添加到本地列表。

主要实现代码如下：

void MyMainDlg::addFolder()//添加文件夹

{

QString dirrr = QFileDialog::getExistingDirectory(this, tr("Open Directory"),"/home", QFileDialog::ShowDirsOnlyQFileDialog::DontResolveSymlinks);

QDir dir(dirrr);

if(!dir.exists())

{

return;

}

dir.setFilter(QDir::Files | QDir::NoSymLinks);

QFileInfoList list = dir.entryInfoList();

int file\_count = list.count();

qDebug()<<file\_count;

if(file\_count <= 0)

{

return;

}

QStringList files;

for(int i=0; i<file\_count;i++)

{

QFileInfo file\_info = list.at(i);

QString suffix = file\_info.suffix();

if(QString::compare(suffix, QString("mp3"), Qt::CaseInsensitive) == 0)

{

QString absolute\_file\_path = file\_info.absoluteFilePath();

qDebug()<<absolute\_file\_path;

files.append(absolute\_file\_path);

}

}

if(files.isEmpty())

return;

int i=0;

while(i<files.count())

{

QString path=QDir::toNativeSeparators(files.at(i));

if(!path.isEmpty())

{

mymaintable->plist->addMedia(QMediaContent(QUrl::fromLocalFile(path)));

}

QFileInfo info(files[i]);

QString m\_name=info.completeBaseName();

QStringList list=m\_name.split("-");

QString songname=list.at(1);

QString singer=list.at(0);

int rowcount= mymaintable->rowCount();

mymaintable->insertRow(rowcount);

mymaintable->setItem(rowcount,0,new QTableWidgetItem(songname));

mymaintable->setItem(rowcount, 1, new QTableWidgetItem(singer));

mymaintable->item(rowcount,1)->setTextAlignment(Qt::AlignRight|Qt::AlignVCenter);

i++;

}

}

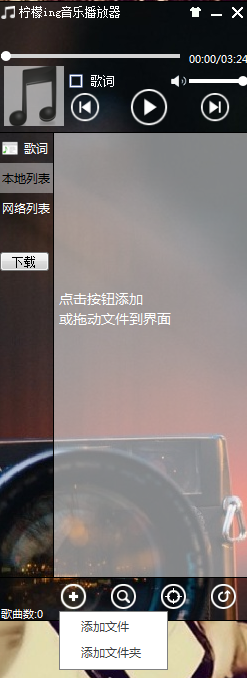


图11 添加文件

### 5.2.2 播放器列表的删除功能

对于歌曲列表中的删除，通过鼠标右击选中歌曲对当前歌曲进行删除或者通过右击菜单可对列表歌曲进行清空。

主要实现代码如下：

void MyMainDlg::delitem()

{

int index= mymaintable->currentRow();

if(index==mymaintable->nowindex)//如果遇到正在播放的歌曲

{

mymaintable->mysetnext();

player->setMedia(mymaintable->plist->media(mymaintable->nowindex));

player->play();

}

mymaintable->mydelitem(index);

mymaintable->removeRow(index);

}

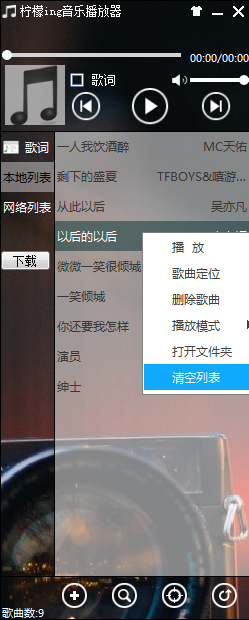


图12 删除歌曲

### 5.2.3 播放器列表的搜索功能

当列表歌曲过多时，当前页面无法全部显示，想要快速查找到想要的歌曲，可通过搜索框对关键字进行检索，其中搜索不限字数，且如果其中有包含一个字的歌曲也都显示出来。

主要实现代码如下：

/\*搜索\*/

void MyMainDlg::setsearchitems(QString words)

{

qDebug()<<words;

int count= mymaintable->rowCount();

for(int i=0;i<count;i++)

{

mymaintable->setRowHeight(i,0);

if( mymaintable->item(i,0)->text().contains(words)|| mymaintable->item(i,1)->text().contains(words))

{

mymaintable->setRowHeight(i,30);

continue;

}

if(words=="")

{

mymaintable->setRowHeight(i,30);

}

}

}



图13 搜索歌曲

### 5.2.4 播放器界面皮肤更换功能

界面皮肤的改变，主要通过信号的方式实现，采用在按键上附上图片，通过点击按键触发信号，界面接收该信号使其达到皮肤更换的目的。

主要实现代码如下：

signalMapper=new QSignalMapper(this);

connect(btn,SIGNAL(clicked()),signalMapper,SLOT(map()));

connect(btn1,SIGNAL(clicked()),signalMapper,SLOT(map()));

connect(btn2,SIGNAL(clicked()),signalMapper,SLOT(map()));

connect(btn3,SIGNAL(clicked()),signalMapper,SLOT(map()));

connect(btn4,SIGNAL(clicked()),signalMapper,SLOT(map()));

connect(btn5,SIGNAL(clicked()),signalMapper,SLOT(map()));

connect(btn6,SIGNAL(clicked()),signalMapper,SLOT(map()));

signalMapper->setMapping(btn,"background3");

signalMapper->setMapping(btn1,"background2");

signalMapper->setMapping(btn2,"background");

signalMapper->setMapping(btn3,"background4");

signalMapper->setMapping(btn4,"background5");

signalMapper->setMapping(btn5,"background6");

signalMapper->setMapping(btn6,"limeng");

connect(signalMapper,SIGNAL(mapped(QString)),this,SIGNAL(skinChange(QString))); connect(signalMapper,SIGNAL(mapped(QString)),this,SLOT(setBk(QString)));

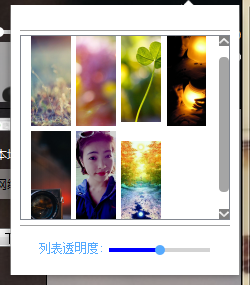


图14 皮肤更改

## 5.3 菜单功能

### 5.3.1 菜单界面

菜单界面主要为循环模式的改变，分为列表循环、随机循环和单曲循环。

主要代码实现如下：

void MyMainDlg::modeAction(QAction \*act)//播放模式

{

if (act == actionCurrentItemInLoop)

{

mynetmaintable->mysetPlaybackMode(1);

mymaintable->mysetPlaybackMode(1);//单曲

}

else if (act == actionLoop)

{

mynetmaintable->mysetPlaybackMode(0);

mymaintable->mysetPlaybackMode(0);//循环

}

else if (act == actionRandom)

{

mynetmaintable->mysetPlaybackMode(2);

mymaintable->mysetPlaybackMode(2);//随机

}

// 设置当前播放模式

}



图15 循环模式

### 5.3.2 菜单功能实现

选中列表中的歌曲，点击鼠标右键，展现出如下图所示菜单，可对当前歌曲进行播放，定位，删除以及打开相应文件夹和清空列表等功能。

主要代码实现如下：

void MyMainDlg::ShowContextMenu(const QPoint& pos)

{

QPoint globalPos = mymaintable->mapToGlobal(pos);

myMenu=new QMenu(this);

myMenu->setContentsMargins(1,1,1,4);

QAction \*play=new QAction(" 播 放",this);

QAction \*songitem=new QAction(" 歌曲定位",this);

QAction \*delsong=new QAction(" 删除歌曲",this);

QAction \*playmodes=new QAction(" 播放模式",this);

QAction \*opensysfile=new QAction(" 打开文件夹",this);

QAction \*emptyList=new QAction(" 清空列表",this);

playmodes->setMenu(playmode);

if( mymaintable->itemAt( mymaintable->mapFromGlobal(QCursor::pos()))!=NULL)

{

myMenu->addAction(play);

myMenu->addAction(songitem);

myMenu->addAction(delsong);

myMenu->addAction(playmodes);

myMenu->addAction(opensysfile);

}

myMenu->addAction(emptyList);

connect(delsong,SIGNAL(triggered()),this,SLOT(delitem()));

connect(emptyList,SIGNAL(triggered()),this,SLOT(emptyList()));

connect(play,SIGNAL(triggered()),this,SLOT(tablemenuplay()));

connect(songitem,SIGNAL(triggered()),this,SLOT(scrolltoitem()));//右键的歌曲定位

connect(opensysfile,SIGNAL(triggered()),this,SLOT(opensysfile()));

myMenu->exec(globalPos);

}



图16 菜单显示

## 5.4 播放设置界面

### 5.4.1 歌词显示实现

歌词显示有两种形式，一种显示为桌面歌词，通过遮罩配合文字的输出，通过结合时间事件函数对其歌词进行解析并跟词逐字滚动；另一种则是通过列表的形式显示歌词，列表中解析过的歌词通过结合时间事件函数，达到逐句滚动的效果。

主要代码实现如下：

void MyLrc::start\_lrc\_mask(qint64 intervaltime)

{

qreal count = intervaltime / 30;

lrc\_mask\_width\_interval = 800 / count;

lrc\_mask\_width = 0;

timer->start(30);

}

void MyMainDlg::positionChanged(qint64 position)

{

ui->horizontalSlider->setValue(position);

QTime currentTime(0, (position/60000)%60, (position/1000)%60); QString format = "mm:ss";

currentTime.toString(format);

QString todisplay;

todisplay = currentTime.toString(format);

ui->label\_playtime->setText(todisplay+"/"+totaldisplay);

qint64 total\_time\_value = player->duration(); if(!lrc\_map.isEmpty())//如果歌词不是空的

{

qint64 previous = 0;

qint64 later = 0;

foreach (qint64 value, lrc\_map.keys())//查找

{

if (position >= value)//如果匹配

{

previous = value;

QTextDocument \*doc=lrcshowedit->document();

QTextCursor textCursor=lrcshowedit->textCursor();

//previous先前的

int positionspre=doc->findBlockByNumber(prviouslrc).position();

textCursor.setPosition(positionspre,QTextCursor::MoveAnchor);

lrcshowedit->setTextCursor(textCursor);

if(!textCursor.hasSelection())

textCursor.select(QTextCursor::Document);

QTextCharFormat formatpre;

formatpre.setForeground(Qt::white);

textCursor.mergeCharFormat(formatpre);

int nowlrc=lrc\_map.keys().indexOf(previous); int positions=doc->findBlockByNumber(nowlrc).position(); textCursor.setPosition(positions,QTextCursor::MoveAnchor);

lrcshowedit->setTextCursor(textCursor);

if(!textCursor.hasSelection()) textCursor.select(QTextCursor::BlockUnderCursor);

QTextCharFormat format;

format.setForeground(QBrush(QColor(250,0,0)));

textCursor.mergeCharFormat(format);

textCursor.movePosition(QTextCursor::Down,QTextCursor::MoveAnchor,12);

lrcshowedit->setTextCursor(textCursor);

prviouslrc=nowlrc;

}

else

{

later = value;

break;

}

}

if (later == 0)

later = total\_time\_value;

QString current\_lrc = lrc\_map.value(previous);

if(current\_lrc != lrc->label->text())

{

lrc->label->setText(current\_lrc);

qint64 interval\_time = later - previous;

lrc->start\_lrc\_mask(interval\_time);

}

}

}

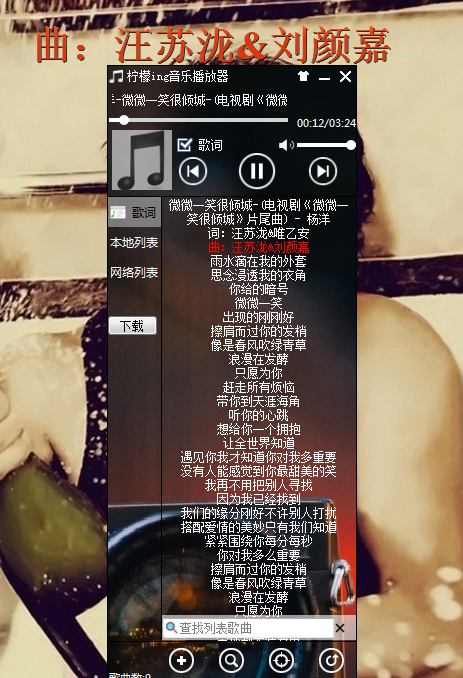


图17 歌词显示

## 5.5 网络下载功能

### 5.5.1 连接服务器

当前服务器界面为空白界面，主要利用TCP协议，通过打开数据库的歌曲文件，以达到向客户端传输网络下载歌曲的目的。

主要代码实现如下：

void Widget::startTransfer()

{

tcpClient = tcpSendServer->*nextPendingConnection*();

qDebug() << "开始发送!!!";

openFile();

qDebug() << "create is success!!!";

qDebug() << fileName;

localFile = new QFile(fileName);

if (!localFile->*open*(QFile::ReadOnly)) {

qDebug() << "client: open file error!";

return;

}

qDebug() << "file is success!!!";

// 获取文件大小

totalBytes = localFile->*size*();

QDataStream sendOut(&outBlock, QIODevice::WriteOnly);

sendOut.setVersion(QDataStream::Qt\_4\_0);

QString currentFileName = fileName.right(fileName.size() - fileName.lastIndexOf('/')-1);

sendOut << qint64(0) << qint64(0) << currentFileName;

totalBytes += outBlock.size();

sendOut.device()->*seek*(0);

sendOut << totalBytes << qint64((outBlock.size() - sizeof(qint64)\*2));

bytesToWrite = totalBytes - tcpClient->write(outBlock);

outBlock.resize(0);

connect(tcpClient, SIGNAL(bytesWritten(qint64)),this, SLOT(updateClient(qint64)));

qDebug() << "发送成功!!!";

}

//读取客户端发送过来的信息

void Widget::readMsgFileName()

{

QString str = tcp->readAll();

qDebug() << str;

fileName = getFilePath(str);

qDebug() << getFilePath(str);

tcp->write("\*");

}

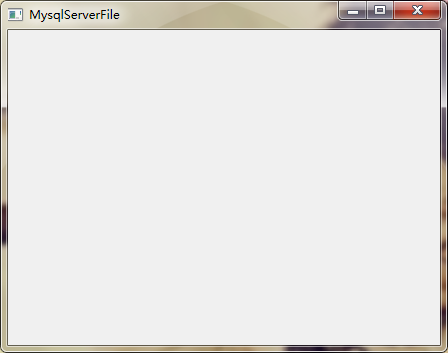


图18 服务器端

### 5.5.2 下载功能的实现

当前客户端下载界面，选取要下载的歌曲文件，通过双击当前要下载的歌曲，连接服务器进行下载，下载之后的歌曲统一存储到文件夹D:\ExcellentMusic中，之后点击导入按钮添加到网络列表中，实现下载功能。

主要代码实现如下：

//通过按钮添加到网络列表

void MyMainDlg::on\_pushButton\_clicked()

{

int i = 0;

mynetmaintable->setRowCount(0);//删除前面的，更新列表

mynetmaintable->clearContents();

QDir dir(DOWNPATH);

QString songname,singer;

QFileInfoList file\_list = dir.entryInfoList(QDir::Files | QDir::Hidden | QDir::NoSymLinks);

while(i < file\_list.size()){

QString m\_name = file\_list.at(i).completeBaseName();

QString path = QString("%1\\").arg(DOWNPATH) + m\_name + QString(".mp3");

qDebug() << path;

if(!path.isEmpty())

{

netplayList->addMedia(QMediaContent(QUrl::fromLocalFile(path)));

qDebug() << QUrl::fromLocalFile(path);

mynetmaintable->plist->addMedia(QMediaContent(QUrl::fromLocalFile(path)));

}

if(m\_name.contains("-"))//如果遍历的时候包含了"-"这个字符

{

QStringList list=m\_name.split("-");

songname=list.at(1);

singer=list.at(0);

}

else

{

songname=m\_name;

singer="";

}

int rowcount= mynetmaintable->rowCount();

mynetmaintable->insertRow(rowcount);

mynetmaintable->setItem(rowcount,0,new QTableWidgetItem(songname));

mynetmaintable->setItem(rowcount, 1, new QTableWidgetItem(singer));

mynetmaintable->item(rowcount,1)->setTextAlignment(Qt::AlignRight|Qt::AlignVCenter);

i++;

}

Widget \*widget = Widget::getInstance();

widget->setGeometry(x()+width(), y()， widget->width(),widget->height());

widget->show();

//// qDebug() << QFileDialog::getExistingDirectory(this,initialName);

}

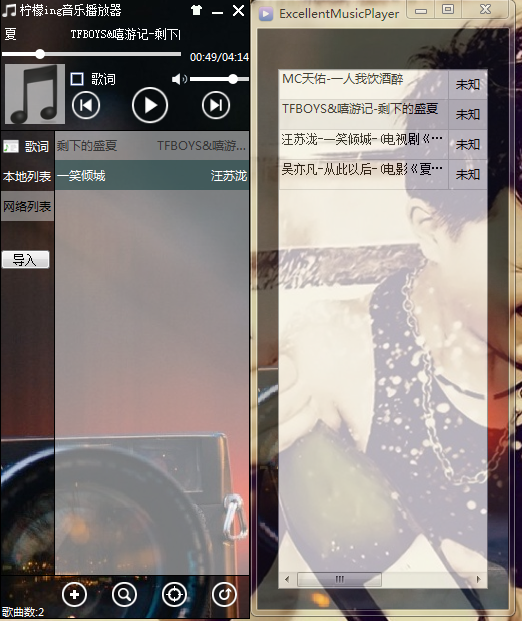


图19下载界面

## 5.6 数据存储方式

### 5.6.1 MySQL数据库存储

本系统设计采用Mysql数据库对歌曲文件进行存储，通过编写数据库脚本导入到Mysql数据库中，最终导入的歌曲文件被服务器所调用传输到客户端，以完成音乐播放器网络下载的功能。

具体文件在数据库中的存储方式如下图20所示。

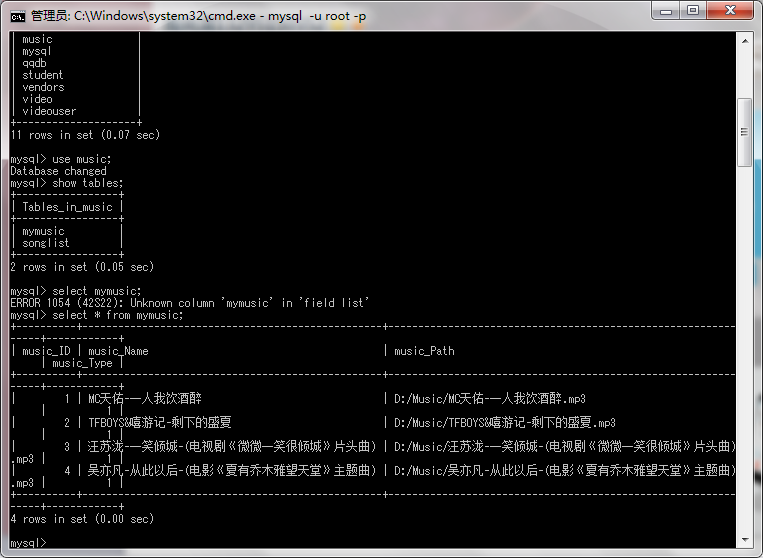


图20 数据库存储

# 6 系统测试与性能分析

## 6.1 系统测试

以下是对该音乐播放器整体设计的测试运行结果与截图：

（1）主界面显示正常，歌曲列表导入成功。见图21。



图21 主界面显示

1. 将文件添加到列表中，可选择两种添加方式，一种是直接添加歌曲文件，另一种则是添加整个歌曲文件夹。见图22。

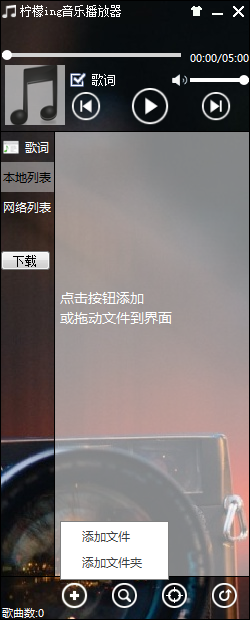


图22 添加歌曲到本地列表中

（3）点击列表歌曲右键，可选择对该歌曲或列表进行操作。见图23。



图23 对列表中的歌曲进行右键设置

（4）点击歌曲定位图标，可对当前播放歌曲进行高亮显示。见图24。

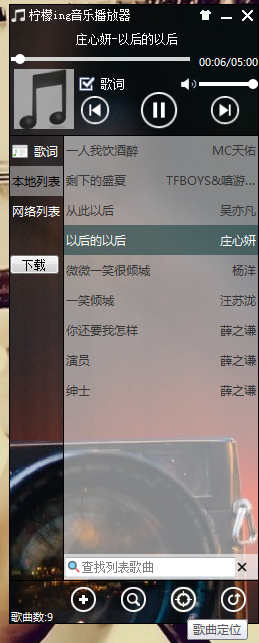


图24 对当前播放歌曲进行定位

1. 桌面歌词跟随歌曲进度成功显示，并且颜色跟随渐变。见图25。



图25 显示桌面歌词

1. 在列表中显示歌词，采取逐句滚动的方式跟随歌曲进度。见图26。

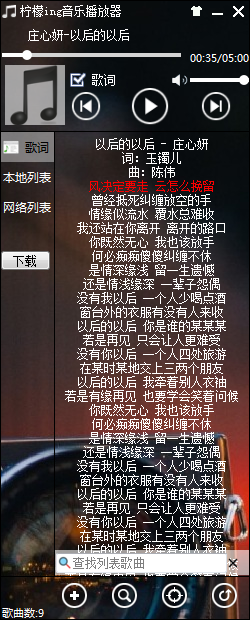


图26 显示列表歌词

（7）列表中的歌曲过多时，为了更快的找到想要听的歌曲，可以在搜索框中输入歌曲名

字中任意一个字进行查找。见图27。



图27 通过搜所找到歌曲

1. 歌曲的播放模式可以点击右下方图标进行更换调节。见图28。



图28歌曲播放模式

（9）播放器小托盘模式里会显示一些常用到的功能键，这样就可以随意切换，十分方便。见图29。



图29 最小化小托盘模式

1. 服务器列表的歌曲文件是通过数据库导入的，双击服务器列表中的歌曲即可下载到客户端的网络列表中，这时，点击导入按键，即可刷新网络列表中的下载数据。见图30。

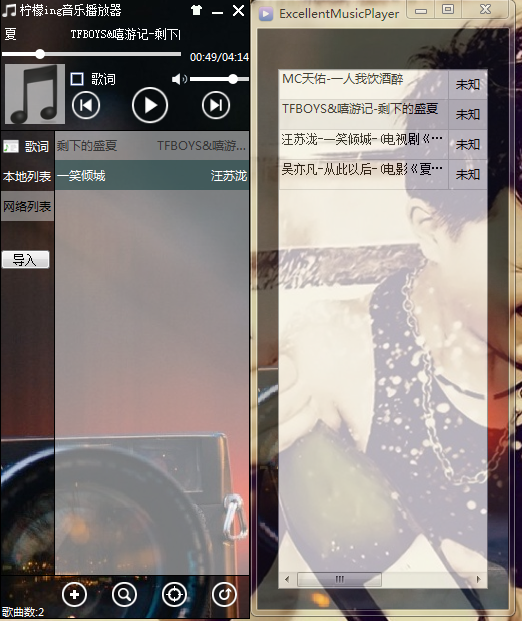


图30 服务器客户端下载界面

## 6.2 性能分析

本系统开发耗资少，符合当代需求，界面简单易操作。在保存歌曲的时候，采用单例模式，不但能减少内存消耗，更大的提高了效率，而且总体性能良好，更好的满足用户需求。

# 结论

本项目设计包括了客户端和服务器端，使用了QT编程，用到了Mysql数据库、文件的读取和操作、多媒体相关功能的知识和实现。通过对这些方面资料的查阅和学习，积累了许多经验，对之后的学习有很大价值。

在本次项目设计过程中，发现了有很多不足，例如当前实现的功能比较简单，在网络下载功能上还没有完善，只是创建服务器与客户端，实现相关文件传递，之后还需要对代码的可读性与网络部分进行优化。在之后的设计中，更好的完善相关功能，并对现有的功能进行优化，然后多方吸取市面上音乐播放器的功能特点，对本设计进行改善。

# 参考文献

[1]李宋琛主编，Linux面向对象窗口高级编程，科学出版社。

[2]孙永杰，引人注目的嵌入式Linux[J]；微电脑世界；2000年35期。

[3]霍亚飞，QT Creator快速入门[M]，北京；北京航空航天大学出版社，2012。

[4]郑阿奇，Visual C++实用教程（第4版）[M]，北京；电子工业出版社，2012。

[5]张红艳译，QT5 开发实战；人民邮电出版社。

[6]明日科技，VisualC++管理信息系统完整项目实例剖析；人们邮电出版社，2005。

1. [↑](#endnote-ref-0)