# 基于 QT 的音乐播放器设计与实现

摘要：计算机技术的飞速发展大大提高了人们的工作效率，　尤其是互联网技术更是很大程度上丰富和方便了人们的生活。　近些年来，人们的生活水平也在不断提升，　在物质丰富的条件下，　人们开始在工作之余关注娱乐，　期望在其它方面释放工作压力，　同时培养自己的兴趣爱好，　随之而来的是人们对多媒体应用的关注，　许多电影播放平台、　音乐播放软件等逐渐深入人们的生活，并慢慢地成为人们生活重要组成部分。

目前，互联网上已经拥有大量的音乐播放软件，　这些软件是各个软件供应商的商品关键组成部分，　当前其实现技术较复杂，　在功能方面相当完备且强大，　如何简单、高效、方便地设计实现一款小巧美观的音乐播放器软件成为人们关注的热点。　基于该问题，　本文设计并实现了基于　Qt 的音乐播放软件，该软件能够便捷、高效地为用户展示音乐播放界面，方便的进行歌曲播放和控制功能。　本文的工作分为软件界面设计和软件功能实现两部分，　其中，　软件界面设计工作主要包括用户界面设计实现；　各个功能模块实现工作主要包括后台程序编码设计模块化完成设计等内容。

该音乐播放软件以　Qt 开发平台中实现歌曲播放的相应功能为基本框架设计，通过C++语言编程实现各个功能函数，软件设计方面采用模块化的软件设计思想实现，具有友好的用户交互界面和高承载能力的运行稳定性。

关键词 ：Qt; Qt Creator; 音乐播放器

**Abstract:** The rapid development of computer technology has greatly improved the efficiency of people's work, especially the Internet technology is to a large extent, rich and convenient for people's lives. In recent years, people's living level also in the continuous upgrading, under the condition of material wealth, people began to in the remaining of the work focused on entertainment, expect to release work pressure, on the other hand, and cultivate their interests and hobbies, the attendant is concern on the application of multimedia, many movies broadcast platform, music player software such as gradually deep into people's life and slowly become an important part of people's lives.

At present, the Internet has a lot of music player software, the software is part of various software vendors of key commodities and the implementation technology is more complex, the function is quite complete, and the powerful, how simple, efficient and convenient to design and implement a clever little beautiful music player software become the focus of people's attention. Based on this problem, this paper designs and implements the music player software based on Qt, which is convenient and efficient for the user to display the music player interface. For software interface design and software implementation of the work division. Among them, software interface design work mainly includes user interface design and implementation; each function module realization mainly includes the backstage code module design to complete the design.

The music player software to Qt development platform to achieve the song playing the corresponding function as the basic framework for the design, using C + + programming language implementation of all functions, software design using modular design idea of the software, with a friendly user interface and high load carrying ability, the stability of operation.

**Keywords:** Qt; Qt Creator; musicplayer

# 目　录

# 一、引　言

随着现代化建设的不断深入和人民生活水平的日益提高，　大量的计算机设备和复杂网络信息系统在各行各业当中广泛布置，　这些系统在完成了原来许多人力才能实现的工作的同时，由于改进了工作模式和工作方法，　使得相应领域的工作效率也迅速提升，　在相等的时间容量里产生了更大的经济羊效益和社会效益。　因此，人们对计算机计算、网络技术和现代通信技术等大量技术在日常生活中的进一步应用产生了关注，　期望在各领域内的实现工作的计算机化、网络化和自动化，提升工作效能。

随着计算机技术和网络技术的迅速发展以及在各个领域的广泛普及，　各行各业的工作都变得信息化、现代化和智能化，这些技术在推动经济持续发展的同时，　也给人们的生活带来了极大的便利，为人们生活水平的提高起到了巨大的推动作用。　科技在日新月异的发展的同时，人们生活质量也在不断的提高，人们工作之余的业余生活也越来越丰富多彩，这些娱乐活动一方面可以减轻人们的工作压力，　增加一些娱乐时间，另一方面，许多人可以在这些娱乐活动当中培养自己的特长和兴趣爱好等，　而音乐正是许多人所共同拥有的一项爱好之一。

在个人计算机快速普及的情形下，各种各样的计算机应用程序层出不穷，纷杂多样，一些领域软件的开发和普及，极大的提高人们的工作效率，让人们在各种软件的帮助下，更加方便快速的完成各项工作任务，而不用像以往那样需要复杂的人工过程。　与此同时，这些软件极大的丰富了人们的娱乐生活，　让人们的娱乐方式更加多样化。　音乐播放软件就是这些众多类型软件中的一种音乐播放器通常运行于个人电脑端，　为用户提供播放音乐的功能，同时也有一些音乐播放软件提供其它的功能如时事资讯等，　音乐播放软件的出现为人们的工作生活带来了乐趣，　提升了用户的娱乐体验，是一种有巨大实用价值的计算机应用程序。

## 1.1背景与意义

计算机的快速发展已经使得人们可以通过更加丰富的手段来获取信息，　传统的情形下，人们大多使用文字来传递信息，　这种信息传递方式具有方便熟悉的特性，　沿用了上千年至今。如今，在计算机技术的巨大推动下，　许多其它的信息传递方式也越来越多的被人们使用着，越来越多的走进了人们的日常生活之中。　而多媒体技术正是这种情形下

产生的一种信息传递技术，我们通常所说的 "媒体"（Media）包括其中的两点含义。一是指信息的物理载体， 这种载体是实实在在的物质存在， 可以将信息对物质方式改变而在载体上留下痕迹，这些载体至今已经沿用了许多年，这类载体包括书本、光盘等；另一层含义是指信息以其外在的表现，给人以感观的形式来传播信息，包括文字、声音、图像等。对于这里所说的多媒体计算机而言， 其主要是指后者， 这种多媒体计算机可以处理文字、图像和动画之类的信息。

多媒体计算机作为目前广泛使用的计算机设备， 其重要功能就是对多媒体文件的播放功能，其中包括的一种格式的多媒体文件就是音乐。 在音乐播放器成为人们了广泛应用的计算机应用软件之后， 人们的日常生活被极大地丰富， 越来越多的人们开始使用音乐播放器来对计算机设备上的音乐文件播放， 而互联网上的音乐目前也正以极大极丰富的产量在生产之中， 每天都会有数以万计的新的音乐产生， 而作为为用户需求考虑的软件开发人员来说， 设计并实现一款音乐播放器应用程序则成为一种应用需求。 目前， 互联网上已经拥有大量的音乐播放器， 这些播放器不仅使用方便快捷， 而且往往拥有强大的功能，并且拥有十分友好的用户交互界面，广受用户的好评。但是用 Qt Creator 开发的音乐播放器小巧而功能齐全， 方便移植到嵌入式平台下或其他平台下， 只需一次编译就可在不同平台下运行播放。 随着现在科技的发展， 越来越多的嵌入式设备已经被广泛应用于生活中，因此嵌入式软件的开发对于嵌入式系统设备的发展有着非常重要的意义。

## 1.2国内外研究现状

计算机的快速发展使用， 让个人计算机迅速在广大人群当中普及开来。 而随着越来越的的个人计算机连接到互联网上， 网络的规模越来越大， 互联网上的资源也越来越丰富多样，各种应用层出不穷，令人目不暇接。而早在互联网开始普及之初，大量的计算机应用就已经开始被人们关注并开始进行开发， 早期的计算机体积大， 存储量小， 可以安装运行的计算机应用程序极为有限，而随着计算机 CPU 和内存储器的质量和规模不断扩展，更多的应用程序出现在互联网上，类型也越来越多样化。

在这一发展趋势下， 互联网娱乐软件的发展也走进了一个新天地。 大量的娱乐软件被世界各地的用户下载使用， 这些应用极大的方便了人们的学习工作和生活， 提升了工作效率， 减轻了工作压力， 也丰富了工作之余的个人生活。 音乐则作为这众多类型的应用当中的一种，已经被人们广泛的使用开来。

最早用来播放音乐的设备应当属于硬件设备， 因为硬件设备出现的相对较早， 而软

件设备的出现是在个人计算机出现之后，　才开始大量的在互联网上出现并运用。　到目前为止，软件播放设备的种类数量和使用量反而超过了硬件播放设备，　因为其具有方便快捷且易携带等特点。目前互联网止广泛应用的音乐播放设备，比较著名的包括　KuGou 音乐播放器，酷我音乐播放器，这些都是专门用于音频媒体文件的播放；　也有一些软件附带的功能中，也包含音乐播放软件，如腾讯　QQ 软件附带的 QQ 音乐播放器，该播放器随着腾讯 QQ 的普及也迅速普及开来，用户数量巨大。

而随着个人计算机设备在人群中的广泛普及，　每个人都可以学习计算机技术、程序设计语言等知识，来开发适合个人使用的，结合自身需求的计算机应用软件。　本文正是在考虑目前互联网上使用量较多的音乐播放器后，试图通过　Qt 开发出一款适合个人使用的音乐播放器软件，能够小巧方便的运行的个人计算机上。

## 1.3研究目标及内容

当今社会，生活水平虽然在不断提高，　但是人们也生活在巨大的压力之下，　需要有能够释放压力的娱乐活动，　听歌则是一种相对比较温和的娱乐方式。　在音乐播放器软件大量推广及应用的情形下，对音乐播放器软件的进一步深入的开发和改进成为人们对音乐播放器软件的一个重要的功能需求。音乐播放器作为人们娱乐生活的重要组成部分，对软件的用户体验与一般软件相比要高，而且对软件的功能需求也要求相对　较多，因此，本文在综合考虑各种可能的功能需求以后，设计实现一个基于　Qt 的音乐播放器软件，实现人们日常音乐播放的一般功能，丰富人们的娱乐生活。

本文所设计实现的软件，是在广泛的研究目前已经正在使用的音乐播放软件之后，吸收了大多数软件的优点和长处，　并拥有自身的界面特点，　针对用户实际使用当中可能会有的功能需求，进而对音乐播放软件进行开发，因此，该软件具有较强的实用性，该软件概括起来具有以下几个方面的特点。

（1）先进的模块化开发思想，对于歌曲的播放、播放控制等各个部分分别对应不同的模块来进行开发，这样既方便了软件的结构设计，也最大程度上提升开发的效率。

（2）　提供良好的人机交互界面，使用　Qt 开发的软件共有的特点之一就是能够开发出具有很好的操作界面的应用程序，这也是本文所开发的软件的优点之一。

（3）可以对歌曲内容进行选择，查找，控制播放等功能，运行方便快捷，可靠性高。

软件的总体开发环境为　Windows 环境，具体使用的编程语言则是采用　C++编程语言，在 Qt Creator 开发平台的环境下进行软件程序的编写。该平台作为一个操作简便的

开发环境，其最大　的优势就是对用户界面应用程序的开发，可以迅速的开发出界面友好、功能完备的应用程序，在良好的用户界面的展示下，用户会拥有更高的操作效率，同时在良好的用户界面下还能够实现软件应当实现的功能。　　总之，良好的界面与功能的结合是该软件的一大优势，基本上可以满足大多数用户的使用需求。

本文在设计实现基于　Qt 的音乐播放器软件的同时，　充分考虑了　Qt 的语言特性和开发环境，发挥了其在用户界面应用程序开发当中的优势，　对音乐播放器软件进行了开发。本软件结合实际使用当中可能使用到的软件功能，并采用软件工程中模块化的开发思想，完成该音乐播放器软件的开发，最终设计并实现一个　Windows 平台下运行的音乐播放器软件。

## 1.4可行性分析

### 1.4.1 经济可行性

随着计算机的普及，　越来越多的人学会了使用计算机。　与此同时，　计算机的价格相对于过去来说也便宜了很多，　但是在运行性能上却有了很大的提升。　在目前互联网上已经广泛存在大量的音乐播放器软件的同时，　开发出具有自身特色的音乐播放器软件，　能够让用户体验到更新的软件体验，　能够极大地减轻用户的长期使用某个软件产生的心理负担，提高用户工作活动的效率。但是，在开发的过程当中仅仅使用到了计算机设备，并没有使用到其它的一些资源，因此，系统在经济上是可行的。

### 1.4.2 技术上可行性

本课题使用　Qt Creator 作为本软件的开发工具。　相比于　VC++，Qt Creator 能够提供更多的面向用户界面调用函数，　在开发界面上也更加人性化，　方便用户使用。　其代码模块化程度非常高，　如果系统日后有一些需要改动的地方，　用此开发工具可以方便地实现对系统的扩展和修改。　因此，本系统在开发上的各种技术条件都是满足的。　它在技术上是可行的。　本系统为一个小型的音乐播放器软件，　它所需要消耗的资源非常小，　而且运行成本低，　一般个人的计算机的硬件条件都能够满足本软件的运行。　所以，　本软件在运行上是可行的。

## 1.5论 文组织结构

本文共分为五章，各章内容如下：

第一章为引言， 首先介绍了课题的研究背景和研究内容， 对音乐播放器的国内处研究现状进行了深入的探讨， 然后阐明了研究的目的和内容， 最后给出了文章的组织结构。

第二章为相关技术综述，该部分重点介绍了开发该软件时所用到的 Qt 程序设计语言，以及本文在程序设计时使用的软件开发环境 Qt creator，对该软件平台的基本情况进行了详细的介绍。

第三章为软件分析与设计， 首先系统的总体结构要求归纳出系统的功能需求， 在需求分析过程当中， 考虑音乐播放器软件所有可能的功能需求， 包括功能完备性、 稳定性、可维护性以及可扩展性，然后根据软件需求分析的结果对软件基本架构进行了设计工作。

第四章为系统主要功能实现，采用模块化的设计思想，利用 Qt creator 软件进行音乐播放器进行设计和功能实现， 实现了较好的用户界面和较快的操作效率， 实现了音乐播放器的基本功能，包括音乐播放功能、暂停功能、播放进度展示功能等一系列功能。

第五章为总结与展望，总结了基于 Qt 的音乐播放器软件的功能特点，对软件的工作流程进行了归纳整理， 同时分析了该音乐播放器的应用前景， 最后对前期的毕业设计工作进行了总结。

# 二、相关技术研究

## 2.1 Qt 介绍

Qt 作为一个应用程序开发框架，可以方便的开发 C++的图形用户界面，另外， Qt 中包装了一组可供调用的 GUI 类，这类的在运行效率高，对于 Qt 的程序的运行速率有很大的帮助。 Qt 作为一个开源的 C++工具包，吸引了众多的开发人员使用 Qt 来进行用户界面应用程序的开发， Qt 也为开发人员提供了极为方便的平台开发工具。

## 2.2 Qt 的优势

Qt Creator 是跨平台的 Qt IDE，Qt Creator 是 Qt 被 Nokia 收购后推出的一款新的轻量级集成开发环境（ IDE）。此 IDE 能够跨平台运行，支持的系统包括 Linux（32 位及 64 位）、 Mac OS X 以及 Windows。根据官方描述， Qt Creator 的设计目标是使开发人员能够利用 Qt 这个应用程序框架更加快速及轻易的完成开发任务。

在功能方面， Qt Creator 包括项目生成向导、高级的 C++ 代码编辑器、浏览文件及类的工具、集成了 Qt Designer、 Qt Assistant、 Qt Linguist、图形化的 GDB 调试前端，集成 qmake 构建工具等。

Qt Creator 主要是为了帮助新 Qt 用户更快速入门并运行项目，还可提高有经验的 Qt 开发人员的工作效率。 使用强大的 C++ 代码编辑器可快速编写代码 ，语法标识和代码完成功能输入时进行静态代码检验以及提示样式上下文相关的帮助代码折叠括号匹配和括号选择模式高级编辑功能。

使用浏览工具管理源代码，集成了领先的版本控制软件，包括 Git、Perforce 和 Subversion 开放式文件，无须知晓确切的名称或位置搜索类和文件跨不同位置或文件沿用符号在头文件和源文件，或在声明和定义之间切换。

为 Qt 跨平台开发人员的需求而量身定制 ，集成了特定于 Qt 的功能，如信号与槽 (Signals & Slots)图示调试器， 对 Qt 类结构可一目了然集成了 Qt Designer 可视化布局和格式构建器只需单击一下就可生成和运行 Qt 项目。

与其它的图形用户界面应用程序开发软件相比， Qt 真正做到了面向程序开发人员的界面开发功能， 该功能很容易使用面向对象技术来实现， 并且能够真正的允许程序开

发人员根据自身的需求对其进行扩展，　Qt 与其它几种 GUI 的对比结果如下表：

表 2-1 Qt 与其它 GUI 的对比

| 名称参数 | MiniGUI | OpenGUI | Qt/Embedded |
|---|---|---|---|
| API（完备性） | Win32（很完备） | 私有（很完备） | Qt(C++)(很完备) |
| 函数库典型大小 | 300KB | 300KB | 600KB |
| 移植性 | 很好 | 只支持 x86 平台 | 较好 |
| 授权条款 | LGPL | LGPL | OPL/GPL |
| 系统消耗 | 小 | 最小 | 最大 |
| 操作系统支持 | Linux | Linux,DOS,QNX | Linux |

## 2.3 面向对象开发过程

在当前计算机领域的前沿中，　最热门的是面向对象的软件开发方面。　其中针对面向对象的问题进行求解这一方面是当前最受关注的重要趋势之一。在众多的开发语言中，C++是一种面向对象的开发语言。　因此，C++也同时具备了面向对象的语言的一些优点，比如说：代码能够使开发人员方便地进行代码维护，　代码能够让开发人员进行相应的扩展，且不会让攻击者进行恶意攻击、　特定功能的代码可以形成一个模块，　然后开发人员可以对其进行重用等优点。这些优点对于面向过程的开发语言来说都是不能够实现的。所以，总结了面向对象的编程技术的优点如下：

（1）可管理性，维护简单

开发人员要开发一个面向过程的系统时，　通常的版本管理指的是管理函数和开发系统过程中的全程变量。而在开发的后期，函数可以做出较大的变动，与此同时，全程变量也可以做出相应的变动。　但是对于一个面向对象的程序来说，　开发人员开发出的一个系统是由对象来组成的。　而对象又是由类来生成的，　所以如果想要管理对象，　只需要对类进行管理就可以了。

（2）模块化

对于面向对象的编程语言来说，　模块化是其一个最明显且最基本的特征。　实体在编程语言中，　会被表示成类，　且它和同一名字空间中的相应类能够具有相同的功能。　开发人员在编程过程中，　能够在名字空间中添加一个简单的类，　但同时也不会影响该名字空间中的其他成员中的作用。

（3）可扩充性

开发人员在开发现代应用软件时，对于软件的可扩充性也会提出相应的要求。可扩充性即是如果开发人员在后期需要对软件做出相应修改或扩充时，需要能够很方便地软件代码进行修改。根据开发规范的规定，这种软件的扩充和修改的相应范围不仅要涉及到相关软件的内容，也可以对软件的形式和工作机制进行相应的修改和扩充。

开发人员在设计面向对象的程度时，要注意其应该具备良好的可扩充性。因为编程语言中的类可以根据人类对于相关事物的理解给予它们相应的意义。因此在后期不会做出很大的改动。开发人员可以利用继承的方法对新的类进行添加相关属性的操作。同时，也可以用它生成系统的原型。

（4）代码重用

开发人员在开发系统时，要把握的一个核心思路便是要提高系统的可重用性。面向对象的程度设计在编程时具备四大特点：抽象、封闭、继承、多态等。这四个特点都是围绕着提高系统的可重用性来进行编码的。

在经典的开发系统的过程中，其可重用性主要体现在以下两个方面：

（1）系统开发的类不仅可以被本系统继承和使用，还可以被别人使用。

（2）代码重用的核心就是使要实现代码能够继承。

# 三、软件分析与设计

## 3.1 需求分析内容

“ 需求 ”一词最早的出现，应该是在经济学领域，指一个特定时期内，消费者在某一价格下对一种商品， 愿意而且能够购买的数量。 由于在社会生活中具有普遍意义， “ 需求 ”的概念一经提出，就被迅速而广泛地应用到政治、经济、军事 等各个领域，内涵也不断丰富。但无论应用到哪 个领域， “ 需求 ”内在的、核心的内涵是基本一致的，即是对事物发展前景、 期望的描述， 实质是提示事物当前状态与期望状态， 当前能力与期望能力之间的差距。

一般情况下， “ 需求 ”具有以下基本特性：一是时间性，即需求是某一时间段内的需求，没有时间约定的需求没有音义二是主观性，即需求源自需求提出方的主观意愿；三是客观性， 即任何主观需求都会受到客观的制约，不管需求提出方是否认识到这些制约，其都客观存在。

需求是指必须实现什么的规格说明，它描述了系统的行为、特点或属性， 是在开发过程中对系统的约束。 需求就是人们对系统的主观期望， 真正的需求存在于人们的脑海中，任何文档形式的需求仅仅是一个模型、一种叙述或描述而已。

一般而言，对某人事物进行需求分析过程大概包括如下几个方面：

（1）需求预测。依靠管理者的经验、国内外类似的经验教训，针对系统设计开发等过程中可能出现的需求变更和新需求，进行预测。

（2）变更控制。在对实施过程中，难免会出现需求的变更，因此需要进行变更控制，首先要明确需求的变更， 然后针对变更的必要性和可靠性、 变更所带来的风险进行评估，确定是否进行变更和如何进行变更。 变更控制的结果将导致需求规格说书版本的演变。

（3）版本规划。由管理者根据需求的迫切性、需求实现的因果关系、设计实现和实施的过程特点等， 对版本演变过程进行规划， 形成对需求管理具有指导意义的 “ 路线图 ”。

（4）风险控制。分析设计实现和实施过程和需求变更过程中的风险因素，宣评估可能带来的费用、进度、性能上的风险，为管理者提出风险管理策略，为版本规划提供

依据。

## 3.2 软件需求分析

在设计实现音乐播放器软件的同时， 可以对目前存在的不同类型的音乐播放器进行广泛深入的研究， 查看软件可能需要的需求内容。 因此，需要对具体问题进行具体分析，深入挖掘其需要实现的系统功能， 以方便后面对软件构架的设计工作。 需求分析的过程，是开发人员对音乐播放器工作过程的认识与熟悉的过程， 也是对软件内部工作流程进行计算机建模的过程， 最终目的是通过需求分析了解用户需求实现的功能， 根据用户提出的需求设计好系统的概念模型， 对用户提出的需求进行计算机方法的描述， 并建立相应配套的需求分析文档， 设计好系统的具体实现方案。 一般而言， 设计人员对系统的需求分析过程大体如图 3-1 所示。

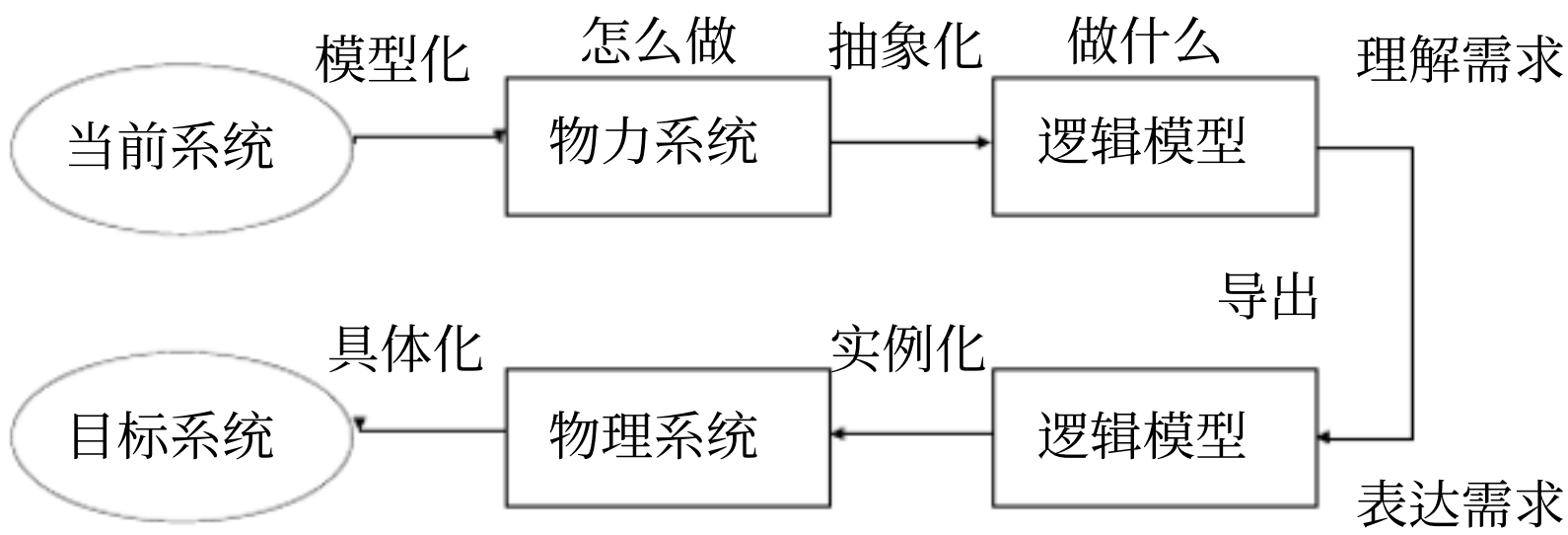


图 3-1 需求分析模型

在设计实现基于 Qt 的音乐播放器的同时， 由于考虑到音乐播放器的实际工作环境，通常音乐播放器工作在个人电脑主机端， 需要考虑到许多方面的特点， 经过本文的分析，可以确定的是该软件应当拥有以下几个方面的特性：

（1） 功能完备性

在当前个人电脑广泛普及的情形下， 电脑端的软件的数量规模和类型复杂多样， 但通常这些软件在设计开发时， 都需要考虑到一个最核心的问题， 那就是软件的功能完备性，这是任何一款想要广泛普及的应用软件必须完成的功能特性， 也是软件使用人员对软件最基本的需求分析。 根据对音乐播放器当前研究现状的分析和探讨， 结合实际生活当中音乐播放器的特点和所需要完成的功能，可以知道， 音乐播放器的设计开发过程当中，应用考虑到音乐播放器需要实现的所有的功能集合， 在进行功能划分的过程当中，可以采用模块化的功能设计思想， 对功能的划分尽可能的细致， 做到不遗漏。 例如音乐播放器应当拥有的最基本的功能：播放音乐，播放控制，音乐列表的显示等。

（2） 稳定性

稳定性是软件设计中一项很重要的指标， 任何一款软件都要拥有很好的稳定性， 该软件才能获得良好的用户体验， 尤其对于娱乐性质的应用软件来说， 用户体验决定了基本上该软件的生命力。 由于音乐播放器在电脑端安装后即开始使用， 其应用将是长期的和持续的。因此，稳定性在软件的设计实现中，显得尤为重要。

（3） 易维护性

音乐播放器的使用是一个长期性的过程， 无论是在主机系统层面上的故障还是音乐播放器软件本身的故障， 都可能影响到软件的使用和用户体验， 因此，需要考虑到软件的易维护性能。 在音乐播放器出现可能的故障问题的同时， 开发人员能够及时发现软件的问题，针对出现的问题进行维护，弥补软件的漏洞。

（4） 可扩展性

可扩展性对于当今的互联网应用软件来说， 同样是一个十分重要的特性。 随着人们消费水平的提升和生活兴趣的广泛拓展， 对于互联网软件应用的功能以及其跟随互联网热点问题的追踪能力十分看重，而对于音乐播放器来说，能够经常性的进行版本升级、定期维护、跟踪时事和流行音乐的更新， 对于用户来说无疑是具有很大的吸引力， 因此，该音乐播放器的开发过程当中，应当充分考虑到软件的可扩展性。

基于以上需求分析， 根据对系统软件的功能特点设计， 本文对音乐播放器软件设计开发的具体细节作了详细介绍。

## 3.3 软件设计

在系统设计阶段， 主要考虑的方面是基于整个系统需要实现的功能， 对物业管理信息系统的整体架构进行科学合理的设计， 使之有一定的规律可以遵循， 不至于进行盲目的设计工作，这对于后期的程序编码和系统的来说意义重大。 通过良好的系统架构设计，使得系统有了一个较好合适正确的数据流和控制流走向， 才能保证音乐播放器的工作过程正规有序， 让用户能够体验到良好的软件质量。 为此，本文在对音乐播放器软件进行软件需求分析之后，针对需求分析的结果，对系统进行了整体的架构设计，如图 3-2 所示：

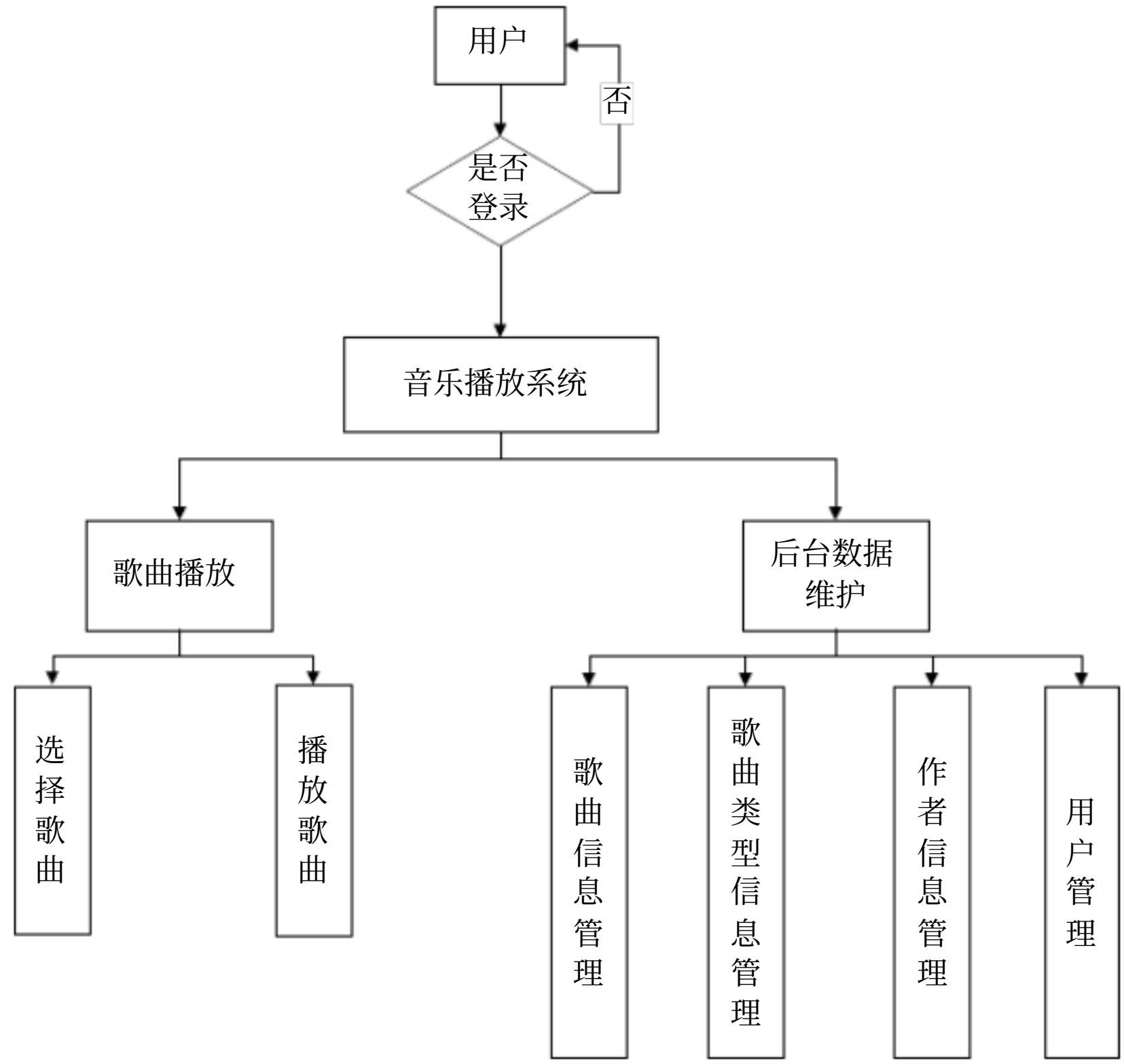


图 3-2 软件模块设计

由上图可以看出， 本文所设计实现的音乐播放器软件， 从软件的架构来看， 主要包括两个功能部分： 软件的歌曲播放部分和后台数据的维护部分。 其中， 歌曲播放部分主要实现该软件的主要功能， 即选择歌曲和播放歌曲； 另一部分则是软件的后台音乐数据的维护，主要在软件的程序代码当中实现， 主要包括歌曲信息管理， 歌曲类型信息管理，作者信息管理和用户信息管理。以上是对该音乐播放器软件的整体工作架构进行的设计，该步骤是后续进一步进行系统功能实现的基础。

# 四、软件功能实现

进入 21 世纪以来，计算机技术飞速发展，它已深深融入到社会生活的方方面面，给人们的工作、 出行方式都带来了翻天覆地的变化， 也为人们的日常娱乐方式带来了广泛的变化。原本旧的计算机软硬件设计已不能适应这快节奏的生活需求， 也阻碍了社会经济的发展。 越来越的人倾向于使用计算机技术来管理自己的工作和生活， 提高自己的工作质量，同时也为生活带来更广泛的娱乐。

本文在设计实现基于 Qt 的音乐播放器软件的同时， 充分考虑了 Qt 的语言特性和开发环境，发挥了其在用户界面应用程序开发当中的优势， 对音乐播放器软件进行了开发。根据前文对该音乐播放器软件的需求分析， 以及对软件的设计分析结果， 结合实际使用当中可能使用到的软件功能， 并采用了软件工程中模块化的开发思想， 完成了该音乐播放器软件的开发，下面分别对软件中重要的功能模块的实现方法及实现效果进行详细的介绍和分析。

## 4.1 软件总体架构

通过在需求阶段对系统的总体功能的分析， 我们得到了这个音乐播放器软件的总体功能结构，其主要功能结构图如图 3-4 所示，它应包括三大基本功能模块。

(1) 选择歌曲文件模块：用户可以方便的查询环境当中存在的歌曲列表信息，并找到满意歌曲的位置和具体的歌曲内容。 此外，歌曲的具体信息可以该文件打开窗口进行更新和修改，以更好地符合实际用户的需要。

(2) 歌曲播放模块：用户根据所选择的具体歌曲让该软件对歌曲进行播放，并且可以实时查看歌曲的播放状态， 对歌曲的播放进行更新和维护， 可以及时处理一些特殊情况，如软件的故障等问题。

(3) 播放状态控制模块：用户可以查看成自己在歌曲播放过程中进度情况，并可以随时更改歌曲的播放进度信息。

## 4.2 软件工作流程

整个软件所设计的工作流程如下：

（1）开始。首先，需要手动打开该软件运行，在 Windows 环境下可以直接方便地

打开该音乐播放器软件，而不需要任何事先的安装与调试工作，这也是该软件的特点之一，目的是实现方便快速的用户操作体验，省略了许多复杂繁琐的安装和调试过程，让软件的工作运行效率更高。

（2）打开文件查找歌曲。在该部分提供最简单方便的选择文件功能，用户需求使用经常在 Windows 环境下采取的选择文件窗口，对所需要的歌曲进行选择查找，查找到目标歌曲后可以点击选中，让软件来进行下一步运行该歌曲。

（3）播放歌曲。该部分是音乐播放器软件最主要的功能部分，播放歌曲的功能实现需要相对复杂的函数调用来实现，在软件开发的过程当中有具体的实现代码，具体可见后续章节。

（4）控制歌曲播放进度。该部分的实现是进一步提高用户的使用体验，方便用户根据其自身的需求对所播放的歌曲进行进度控制，可以暂停歌曲的播放，也可以继续歌曲的播放功能。

（5）结束。点击关闭软件，结束软件运行。

以上整个过程的流程图如下所示：

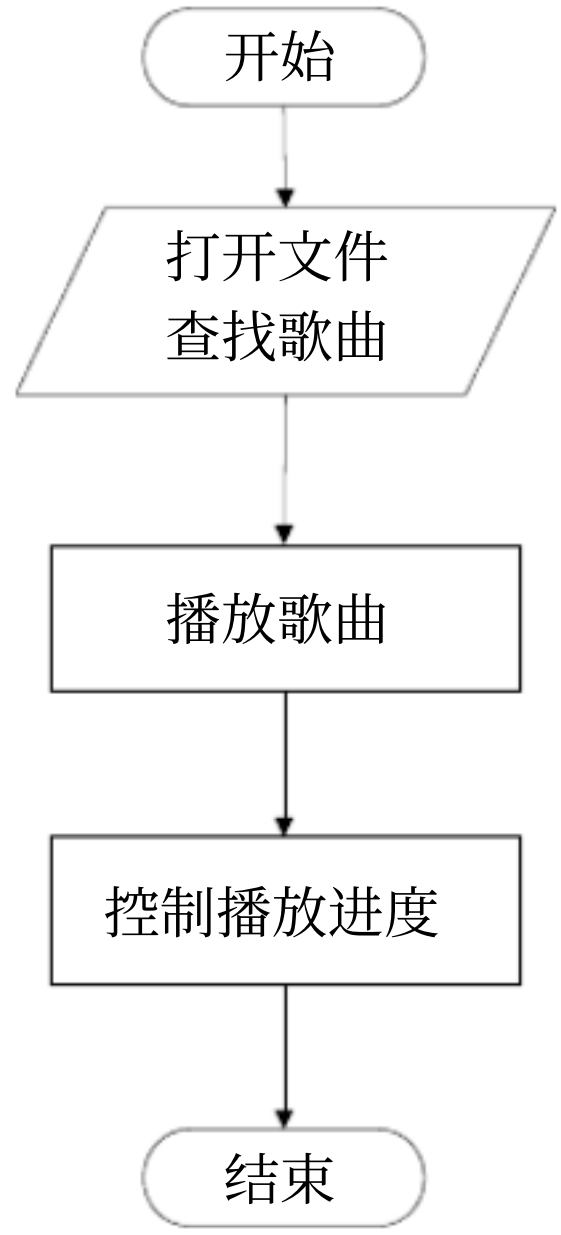


图 4-1 软件工作流程

## 4.3 软件功能实现

软件的总体开发环境为 Windows 环境，具体使用的编程语言则是采用 C++编程语言，在 Qt Creator 开发平台的环境下进行软件程序的编写。该平台作为一个操作简便的开发环境，其最大 的优势就是对用户界面应用程序的开发，可以迅速的开发出界面友好、功能完备的应用程序，在良好的用户界面的展示下，用户会拥有更高的操作效率，同时在良好的用户界面下还能够实现软件应当实现的功能。 总之，良好的界面与功能的结合是该软件的一大优势，基本上可以满足大多数用户的使用需求。

本选题是基于 QT 的音乐播放器，包括音频文件的添加与删除，下一曲，上一曲，播放与暂停，播放模式，显示歌词等。

具体功能如下：

（1）播放状态显示：

用户界面能显示正在被播放的歌曲的进 度，显示歌曲的播放状态；删除歌曲，从音乐播放器中删除选定的歌曲；

用户能够显示正在被播放歌曲的音量；

用户界面能够显示被播放歌曲的歌名。

用户界面能够显示歌词和桌面歌词

（2）播放控制：

用户能够控制正在被播放歌曲的进度；

用户能调节正在被播放歌曲的音量；

用户能够控制歌曲停止、暂停与播放；

用户能够切换上一首、下一首歌曲。

（3）列表中歌曲管理：

在歌曲列表中用户能添加所需的歌曲文件的 文件名，并长期保存在该列表中 ,直到用户 删除该列表中的歌曲名。

（4）核心播放控件：

能够打开 MP3 文件，并将其解码，然后启动音频硬件播放歌曲。

在软件的代码实现中， 本文根据需要实现的功能， 编写了相应的代码， 主要的代码的函数名称和相应的函数声明在头文件 MainInterface.h 中，具体如下：

```
class MainInterface : public QWidget
{
    Q_OBJECT
```

```
public:
    explicit MainInterface(QWidget *parent = 0);
    ~MainInterface();
protected:
    void moveEvent( QMoveEvent * );
    void contextMenuEvent( QContextMenuEvent * );
    void wheelEvent ( QWheelEvent * );
    void closeEvent( QCloseEvent * );
    void dragEnterEvent ( QDragEnterEvent * );
    void dropEvent ( QDropEvent * );
    bool eventFilter  ( QObject *, QEvent * );
private slots:
    void on_toolButton_open_clicked();
    void on_toolButton_List_clicked(bool checked);
    void on_toolButton_playpause_clicked();
    void on_toolButton_stop_clicked();
    void on_toolButton_next_clicked();
    void on_toolButton_previous_clicked();
    void on_toolButton_lrcD_clicked( bool checked);
    void tableWidget_cellDoubleClicked( int row);
    void clearPlayList();
    void setPosition(int value);
    void positionChanged( qint64);
    void audioStateChanged( QMediaPlayer::State state );
    void iconActivated( QSystemTrayIcon::ActivationReason reason);
    void audiolistPositionChanged( int index);
private:
    void readSettings();
    void writeSettings();
```

```
        void loadCurrentLrc();
        void creatActions();
private:
        Ui::MainInterface *ui;
        QTextEdit    *m_text;
        QSlider *m_seekSlider;
        QSlider *m_volSlider;
        QMediaPlayer *m_audio;
        QMediaPlaylist *m_audioList;
        MusicList *m_playList;
        LrcInterface *m_lrc;
        QIcon * iconplay;
        QIcon * iconpause;
        QAction * play , *stop , *open , *sound , *exit , *remove ;
        QSystemTrayIcon *trayicon;
        QMenu *trayiconMenu;
        bool isTouched;
};
class LrcInterface : public QLabel
{
        Q_OBJECT
public:
        explicit LrcInterface(QWidget *parent = 0);
        void setTime(int num){time = num;}
        void setLrcWidth();
protected:
        void mousePressEvent(QMouseEvent *);
        void mouseMoveEvent(QMouseEvent *);
        void contextMenuEvent(QContextMenuEvent *ev);
```

```
        void paintEvent(QPaintEvent *);
private slots:
        void timeout();
public:
                QAction *exit;
                QTimer *timer;
private:
                QPoint dragPosition;
                int time;
                qreal length;
                qreal lrcWidth;
        };
namespace Ui {
        class MusicList;
}
class MusicList : public QWidget
{
        Q_OBJECT
public:
        explicit MusicList(QWidget *parent = 0);
        ~MusicList();
        void listReadSettings();
        void listWriteSettings();
        void clearList();
protected:
        void changeEvent(QEvent *e);
        void closeEvent(QCloseEvent *);
        void moveEvent(QMoveEvent *);
signals:
```

```
        void listClose();
public:
            Ui::MusicList *ui;
        };
```

### 4.3.1 播放歌曲模块

作为该软件最主要， 也是用户使用的最多的功能模块， 歌曲的播放功能决定了该软件能否为用户所广泛的接受， 现代化的软件开发过程中， 最为主要的特点之一， 倒是能够实现以用户需求为目标的软件的核心功能，核心功能的实现决定了软件的后续生命力。因此，该音乐播放器在设计开发过程当中， 把最主要的精力放在了歌曲的播放控制模块，通过大量的相关函数的编写和调用，与 Windows 环境下的音频服务相互交互的过程中，实现音乐播放功能。

首先进入该软件后， 软件的后台代码部分会自动进入软件的主函数， 主函数是软件运行的入口函数，该部分的主函数的具体实现代码如下：

```
int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainInterface w;
    w.show();
    return a.exec();
}
```

播放歌曲的时候需要首先对歌曲目录进行选择， 即打开文件窗口选择歌曲文件的位置，该部分具体的实现代码如下：

```
void MainInterface::on_toolButton_open_clicked()
{
    QStringList urls = QFileDialog::getOpenFileNames(this,tr("open the mp3 file"),
QStandardPaths::writableLocation(QStandardPaths::MusicLocation));
    if(urls.isEmpty()) return;
    int index = m_playList->ui->tableWidget->rowCount();
```

```
for(int i=0; i<urls.length(); i++)
{
    QMediaContent con(urls.at(i).trimmed());
    m_audioList->addMedia(con);
    QString    fileName =urls.at(i).trimmed();
    QString title = fileName.right(fileName.length() - fileName.lastIndexOf('/') -
1);
    m_playList->ui->tableWidget->insertRow(index+i);
    QTableWidgetItem *titleItem = new QTableWidgetItem(title);
    m_playList->ui->tableWidget->setItem(index+i,0,titleItem);
}

m_audioList->setCurrentIndex(index);
m_audio->play();
}
```

歌曲播放功能的具体实现代码如下：

```
MainInterface::MainInterface(QWidget *parent) :
    QWidget(parent),ui(new Ui::MainInterface),isTouched(false)
{
    ui->setupUi(this);
    this->setAcceptDrops(true);
    this->setFixedSize(300,150);
    this->setWindowTitle(tr(" 聆听，播放器 "));
    readSettings();
    m_text = new QTextEdit(this);
    m_text->hide();
    ui->textEdit->setVisible(false);
    m_playList = new MusicList(this);
    m_playList->setFixedSize(300,320);
    m_playList->ui->tableWidget->setColumnWidth(0,300);
```

```
        m_playList->ui->tableWidget->setColumnWidth(1,0);
    connect(m_playList->ui->tableWidget,SIGNAL(cellDoubleClicked(int,int)),SLOT(tabl
eWidget_cellDoubleClicked(int)));
        connect(m_playList,SIGNAL(listClose()),ui->toolButton_List,SLOT(toggle()));
        m_playList->installEventFilter(this);
        m_playList->ui->tableWidget->installEventFilter(this);
        m_volSlider = new QSlider(this);
        m_volSlider->move(190,80);
        m_volSlider->resize(50,20);
        m_volSlider->setStyleSheet("background-color:rgb(255,255,255,100)");
        m_volSlider->setFixedWidth(100);
        m_volSlider->setOrientation(Qt::Horizontal);
        m_volSlider->setRange(0,100);
        m_seekSlider = new QSlider(this);
        m_seekSlider->move(10,35);
        m_seekSlider->resize(170,20);
        m_seekSlider->setStyleSheet("background-color:rgb(255,255,255,100)");
        m_seekSlider->setOrientation(Qt::Horizontal);
        connect(m_seekSlider,SIGNAL(sliderMoved(int)),SLOT(setPosition(int)));
        m_lrc = new LrcInterface(NULL);
        QPalette palette = ui->label_lrc->palette();
        palette.setColor(QPalette::WindowText,Qt::darkBlue);
        ui->label_lrc->setPalette(palette);

        m_audio = new QMediaPlayer(this);
        m_audioList =new QMediaPlaylist(this);
        m_audio->setPlaylist(m_audioList);
        m_audio->setNotifyInterval(10);
        m_volSlider->setValue(m_audio->volume());
```

connect(m_audio, SIGNAL(positionChanged(qint64)), SLOT(positionChanged(qint64)));

connect(m_audio,SIGNAL(stateChanged(QMediaPlayer::State)),SLOT(audioStateChanged(QMediaPlayer::State)));

connect(m_audioList, SIGNAL(currentIndexChanged(int)), SLOT(audiolistPositionChanged(int)));

connect(m_volSlider,SIGNAL(valueChanged(int)),m_audio,SLOT(setVolume(int)));

connect(m_audio,SIGNAL(volumeChanged(int)),m_volSlider,SLOT(setValue(int)));

creatActions();

}

### 4.3.2 播放控制模块

歌曲的播放控制模块包括歌曲的暂停、继续播放、七咲风花切换、选择上一首、选择下一首等功能，这些功能相当于软件辅助功能，围绕歌曲播放这个核心功能而实现。软件开发过程中的主要思想方法就是软件的模块化设计思想，面围绕这一思想，在理论分析层面设计规划的同时，在软件代码实现部分的具体实施中，则是围绕核心功能与辅助功能而实现，具体来说，软件的核心功能是软件最重点的袜，目标，而辅助功能则作为一些软件的细节，同样可以影响用户体验。

软件对歌曲的播放控制功能有以下几部分：控制歌曲的暂停与播放、快进、快退，重新选择歌曲、音量控制等。歌曲播放控制模块的具体实现代码如下：

```
void MainInterface::audioStateChanged(QMediaPlayer::State state)
{
    switch(state)
    {
        case QMediaPlayer::PlayingState:
            {
```

```
m_lrc->setText("Music ...");
ui->label_lrc->setText("Music ...");
loadCurrentLrc();
play->setIcon(*iconpause);
ui->toolButton_playpause->setIcon(*iconpause);
ui->toolButton_playpause->setToolTip(tr("暂停"));
play->setText(tr("暂停"));
QString fileName =
m_audio->currentMedia().canonicalUrl().toString();
QString title = fileName.right(fileName.length() -
fileName.lastIndexOf('/') - 1);
this->setWindowTitle(title);
ui->label_palyname->setText(title);
trayicon->setToolTip(tr(" 播放")+ title);
        break;
    }
case QMediaPlayer::PausedState:
    {
        play->setIcon(*iconplay);
        ui->toolButton_playpause->setIcon(*iconplay);
        ui->toolButton_playpause->setToolTip(tr(""));
        ui->label_palyname->setText(tr("暂停播放 "));
        play->setText(tr("播放"));
        trayicon->setToolTip(tr(" 暂停"));
        m_lrc->timer->stop();

        break;
    }
case QMediaPlayer::StoppedState:
    {
```

```
ui->toolButton_playpause->setIcon(*iconplay);
play->setText(tr("播放"));
play->setIcon(*iconplay);
this->setWindowTitle(tr("Music .."));
        ui->label_palyname->setText(tr("Music .."));
        trayicon->setToolTip(tr("stop"));
        ui->label_time->setText("00:00 / 00:00");
        m_lrc->setText("Music ...");
        ui->label_lrc->setText("Music ...");
        m_lrc->timer->stop();
        break;
    }

default:break;
        }
    }
    void MainInterface::positionChanged(qint64 time)
    {

        qint64 temp = m_audio->duration();
        QTime totalTime(0,(temp / 60000) % 60,(temp / 1000) % 60,time %1000);
        QTime curTime(0,(time / 60000) % 60,(time / 1000) % 60,time %1000);
        ui->label_time->setText(tr("%1
/ %2").arg(curTime.toString("mm:ss")).arg(totalTime.toString("mm:ss")));
        ui->label_time->update();

        m_seekSlider->setRange(0, temp);
        m_seekSlider->setValue(time);
```

```
if(ui->textEdit->find(curTime.toString("mm:ss.zzz").left(7)))
{
    QString str =
ui->textEdit->textCursor().block().text().replace(QRegExp("\\[\\d{2}:\\d{2}\\.\\d{2}\\]"),"");
ui->label_lrc->setText(str);
m_lrc->setText(str);
m_lrc->setLrcWidth();

QTime tt = curTime;
int b = 1,c= 0;
m_text->setText(ui->textEdit->document()->toPlainText());
bool over = ui->textEdit->textCursor().block().next().text().isEmpty();
while(!over
&&!m_text->find(tt.addMSecs(b*100).toString("mm:ss.zzz").left(7)))
    {
        b++;  c++;
    }
    while(over
&&  !m_text->find(tt.addMSecs(b*100).toString("mm:ss.zzz").left(7),QTextDocument::Find
Backward))
    {
        b++;  c++;  break;
    }
        m_lrc->timer->start(c);
}
else
if(ui->textEdit->find(curTime.toString("mm:ss.zzz").left(7),QTextDocument::FindBackward
))
{
```

```
                QString                                str                                =
ui->textEdit->textCursor().block().text().replace(QRegExp("\\[\\d{2}:\\d{2}\\.\\d{2}\\]"),"");
                ui->label_lrc->setText(str);
                m_lrc->setText(str);
                m_lrc->setLrcWidth();

                QTime tt = curTime;
                int b = 1;
                int c= 0;
                m_text->setText(ui->textEdit->document()->toPlainText());

                bool over = ui->textEdit->textCursor().block().next().text().isEmpty();
                while(!over
&& !m_text->find(tt.addMSecs(b*100).toString("mm:ss.zzz").left(7)))
                {
                    b++;    c++;
                }
                while(over
&&   !m_text->find(tt.addMSecs(b*100).toString("mm:ss.zzz").left(7),QTextDocument::Find
Backward))
                {
                    b++;   c++;     break;
                }
                m_lrc->timer->start(c);
            }
        }
```

### 4.3.3 歌曲列表管理模块

播放列表管理模块的主要作用是使得音乐播放器能将歌曲列表实时的展现给用户，方便用户对播放歌曲的信息进行实时查看，以便进一步根据自身需求作出相应的选择。

作为与用户信息交互量大的一个模块，　该模块在具体实现中也作为王菲重要的功能，　为音乐播放器的核心功能服务。　访模块可以纪录音乐播放器的歌曲播放具体列表信息，　增加了软件功能的完备性，　提升了软件的用户体验。　歌曲列表的管理模块具体实现代码如下：

```
void MusicList::listReadSettings()
{
        QSettings settings("test", "List");
        move( settings.value("pos", QPoint( this->x(),this->y()+150) ).toPoint());
}
void MusicList::clearList()
 {
          int count = ui ->tableWidget ->rowCount();
          for( int i=0;i<count;i++ )
                ui->tableWidget->removeRow(0);
          ui->tableWidget->close();
          ui->tableWidget->show();
  }

void MusicList::listWriteSettings()
{
        QSettings settings( "test","List");
        settings.setValue( "pos", pos());
}

        void MusicList::moveEvent( QMoveEvent *e)
        {
                if(qAbs( ( this -> y() – this-> parentWidget()->y() -
this->parentWidget()->frameGeometry().height())) < 20)
                {
                        this->move(this->x(),this->parentWidget()->y() + this ->parentWidget()
```

```
->frameGeometry().height());
            }
            if( qAbs( this->x() + this ->frameGeometry().width()    –this ->parentWidget() ->x())
< 20)
            {

this->move( this->parentWidget()->x()-this->frameGeometry().width(),this->y());
            }

if(qAbs(this->x()-this->parentWidget()->frameGeometry().width()-this->parentWidget()->x(
)) < 20)
            {

this->move(this->parentWidget()->x()+this->parentWidget()->frameGeometry().width(),this-
>y());
            }
            QWidget::moveEvent(e);
      }
      void MainInterface::positionChanged(qint64 time)
      {

            qint64 temp = m_audio->duration();
            QTime totalTime(0,(temp / 60000) % 60,(temp / 1000) % 60,time %1000);
            QTime curTime(0,(time / 60000) % 60,(time / 1000) % 60,time %1000);
            ui->label_time->setText(tr("%1
/ %2").arg(curTime.toString("mm:ss")).arg(totalTime.toString("mm:ss")));
            ui->label_time->update();

            m_seekSlider->setRange(0, temp);
            m_seekSlider->setValue(time);
```

```
if(ui->textEdit->find(curTime.toString("mm:ss.zzz").left(7)))
{
                QString                str                =
ui->textEdit->textCursor().block().text().replace(QRegExp("\\[\\d{2}:\\d{2}\\.\\d{2}\\]"),"");

                ui->label_lrc->setText(str);
                m_lrc->setText(str);
                m_lrc->setLrcWidth();

                QTime tt = curTime;
                int b = 1,c= 0;
                m_text->setText(ui->textEdit->document()->toPlainText());
                bool over = ui->textEdit->textCursor().block().next().text().isEmpty();
                while(!over
&&!m_text->find(tt.addMSecs(b*100).toString("mm:ss.zzz").left(7)))
                {
                        b++;    c++;
                }
                while(over
&&    !m_text->find(tt.addMSecs(b*100).toString("mm:ss.zzz").left(7),QTextDocument::Find
Backward))
                {
                        b++;    c++;    break;
                 }
                m_lrc->timer->start(c);
            }
            else
if(ui->textEdit->find(curTime.toString("mm:ss.zzz").left(7),QTextDocument::FindBackward
```

```
))
        {
            QString str =
ui->textEdit->textCursor().block().text().replace(QRegExp("\\[\\d{2}:\\d{2}\\.\\d{2}\\]"),"");
            ui->label_lrc->setText(str);
            m_lrc->setText(str);
            m_lrc->setLrcWidth();

            QTime tt = curTime;
            int b = 1;
            int c= 0;
            m_text->setText(ui->textEdit->document()->toPlainText());

            bool over = ui->textEdit->textCursor().block().next().text().isEmpty();
            while(!over
&& !m_text->find(tt.addMSecs(b*100).toString("mm:ss.zzz").left(7)))
            {
                b++;    c++;
            }
            while(over
&&  !m_text->find(tt.addMSecs(b*100).toString("mm:ss.zzz").left(7),QTextDocument::Find
Backward))
            {
                b++;  c++;    break;
            }
            m_lrc->timer->start(c);
        }
    }
```

### 4.3.4 软件界面模块

基于 Qt 的音乐播放器的最大优点， 就是该软件可以充分发挥 Qt 编程语言在用户界面应用程序开发上的巨大优势， 这一点也在本音乐播放器的实现上有所体现， 软件界面部分实现的具体代码如下：

```
LrcInterface::LrcInterface(QWidget *parent) :
    QLabel(parent)
{
    this->setWindowFlags(Qt::SubWindow |Qt::FramelessWindowHint|
Qt::WindowStaysOnTopHint);
    this->resize(1024,60);
    this->setText(tr("Music ..."));
    this->setAttribute(Qt::WA_TranslucentBackground);

    this->setCursor(Qt::OpenHandCursor);

    exit = new QAction(tr(" 隐藏 (&D)"),this);
    connect(exit,SIGNAL(triggered()),this,SLOT(close()));

    timer = new QTimer(this);
    connect(timer,SIGNAL(timeout()),this,SLOT(timeout()));

    this->move(400,700);

    lrcWidth = 0;
}

void LrcInterface::mousePressEvent(QMouseEvent *event)
{
    if(event->button() == Qt::LeftButton)
    {
        dragPosition = event->globalPos() - frameGeometry().topLeft();
```

```
        event->accept();
}
else if(event->button() == Qt::MidButton)
        close();
        QLabel::mousePressEvent(event);
}

void LrcInterface::mouseMoveEvent(QMouseEvent *e)
{
        if( e->buttons() & Qt::LeftButton)
        {
                move( e->globalPos() –dragPosition );
                e->accept();
        }
        QLabel::mouseMoveEvent(e);
}

void LrcInterface::contextMenuEvent(QContextMenuEvent *ev)
{
        QMenu menu;
        menu.addAction(exit);
        menu.exec( ev->globalPos());
        QLabel::contextMenuEvent(ev);
}

void LrcInterface::paintEvent( QPaintEvent *)
{
        QPainter painter( this );
        painter.setRenderHints( QPainter::Antialiasing | QPainter::TextAntialiasing);
```

```
QFont font(tr("Times New Roman"),30,QFont::Bold);
painter.setFont(font);
QLinearGradient lg(0, 20, 0, 50);
lg.setColorAt(0 , QColor(0, 170, 255, 255));
lg.setColorAt(0.2 , QColor(61, 214, 191, 250));
lg.setColorAt(0.5 , QColor(85, 255, 255, 255));
lg.setColorAt(0.8 , QColor(61, 214 , 191, 250));
lg.setColorAt(1 , QColor(0 , 170 ,255 , 255));
painter.setBrush( lg );
painter.setPen(Qt::NoPen);
QPainterPath textPath;
textPath.addText(0,50,font,text());
painter.drawPath(textPath);

        length = textPath.currentPosition().x();

        painter.setPen(Qt::yellow);
        painter.drawText(0,14,lrcWidth,50,Qt::AlignLeft,text());
}
```

# 五、结论与展望

## 5.1 软件功能总结

通过在需求阶段对系统的总体功能的分析，进而在软件设计阶段对软件的具体架构和运行流程进行了设计，然后在软件实现方面对软件的功能目标进行了实现，我们得到了这个音乐播放器软件的总体功能结构，其主要功能结构图包括三大基本功能模块。

(1) 选择歌曲文件模块：用户可以方便的查询环境当中存在的歌曲列表信息，并找到满意歌曲的位置和具体的歌曲内容。此外，歌曲的具体信息可以该文件打开窗口进行更新和修改，以更好地符合实际用户的需要。

(2) 歌曲播放模块：用户根据所选择的具体歌曲让该软件对歌曲进行播放，并且可以实时查看歌曲的播放状态，对歌曲的播放进行更新和维护，可以及时处理一些特殊情况，如软件的故障等问题。

(3) 播放状态控制模块：用户可以查看成自己在歌曲播放过程中进度情况，并可以随时更改歌曲的播放进度信息。

本软件的开发环境为 Windows 环境 下使用 Qt Creator 的开发平台，该平台作为一个操作简便的开发环境，可以迅速的开发出界面友好、功能完备的应用程序，该软件可以解决现阶段大多数用户的音乐播放功能需求，并能实现较高的操作效率和较好的用户交互体验，能够满足用户音乐播放功能需求。

## 5.2 软件工作流程

整个软件所设计的工作流程如下：

（1）开始。首先，需要手动打开该软件运行，在 Windows 环境下可以直接方便地打开该音乐播放器软件，而不需要任何事先的安装与调试工作，这也是该软件的特点之一，目的是实现方便快速的用户操作体验，省略了许多复杂繁琐的安装和调试过程，让软件的工作运行效率更高。

（2）打开文件查找歌曲。在该部分提供最简单方便的选择文件功能，用户需求使用经常在 Windows 环境下采取的选择文件窗口，对所需要的歌曲进行选择查找，查找到目标歌曲后可以点击选中，让软件来进行下一步运行该歌曲。

（3）播放歌曲。该部分是音乐播放器软件最主要的功能部分，播放歌曲的功能实

现需要相对复杂的函数调用来实现，　在软件开发的过程当中有具体的实现代码，　具体可见后续章节。

（4）控制歌曲播放进度。该部分的实现是进一步提高用户的使用体验，方便用户根据其自身的需求对所播放的歌曲进行进度控制，　可以暂停歌曲的播放，　也可以继续歌曲的播放功能。

## 5.3 应用展望

本文设计的基于　Qt 的音乐播放器软件吸取了当前主流大型音乐播放器软件的功能优点，并根据本文所需要实现的软件的功能特点，　结合用户的实际需要，　设计出功能齐全、风格简洁且操作简便的音乐播放器软件。　与众多大型播放器软件不同的是，　本文所设计开发的播放器采用　Qt 实现软件的编程开发，其实现过程简单高效，并不需要特殊的编程及设计能力，更适合于普通开发者的设计需要。

在此过程，我学会了许多关于软件需求分析、软件功能设计、软件结构设计以及 Qt 编程开发的相关知识，熟悉了音乐播放器软件的基本工作流程，最重要的是在这个过程当中锻炼了自身认识问题和解决问题的能力素质。

通过几个月的系统开发，　一个音乐播放器软件的所有功能都按照需求分析得到了解决。整个软件从需求到设计完成完全采用了软件工程的设计思想，　前期的每一步设计都是为后期的设计做准备，所以该软件的开发必须在项目开始时就能很好的定位方向。

本音乐播放器软件可以当作小型的台式主机和个人笔记本电脑端供个人使用，　也可以简单的当做某公司或部门的产品显示平台使用，　具有较广泛地适用性和拓展性。　通过这次毕业设计，　我对计算机这个方便人们生活的工具有了新的理解，　相信我会在以后能利用它为人们创造更多的价值。

## 5.4 工作总结

本文所设计的基于　Qt 的音乐播放器软件，　在 Windows 系统环境下可以成功的运行，对于预期的功能目标已经基本实现，其中包括歌曲的选择，歌曲打开，歌曲播放，歌曲暂停，显示歌曲播放的进度条等功能。该软件的开发是在　Qt Creator 的开发环境下完成的程序编写工作，该开发平台简单方便，操作快捷，可以直接在　Windows 系统平台下安装运行，完成代码的编写后，　只需要直接点击编译和运行即可使音乐播放器开始工作，

按用户的选择进行歌曲播放等功能， 并且该软件在开发完成后， 可以跨平台运行， 而需要特别的运行环境配置， 极大的方便了用户的使用过程， 也降低了系统运行的资源开销，具有一定的实用性。

通过本次毕业设计 ,我终于明白了 ‘看一万行代码，不如动手写一行代码 ’这一句真理，对于工科类的学生来说， 除了加强对对书本里的理论知识的学习之外， 更重要的是培养自己实践动手的能力。 这次毕业设计， 让我以后面对困难时变得更有耐心， 对我来说，这一精力都是在以后的生活和学习中的很宝贵的财富， 极大的影响我以后的成长和发展道路。

开发时间限制，我实现了系统的基本功能， 软件可以实现基本的小区物业管理功能，已达到毕业设计的基本要求，但是界面尚不够美观，系统也不够完善，下一步，我会继续改进系统。本次毕业设计以及系统的实现让我认识到了以前很多没有注意到的细节问题，让我学到了不少的新知识。 比如在系统的界面设计过程中， 不仅要在功能上做到没有纰漏， 也要延长系统生命周期， 易于维护与管理， 所以尽管界面设计并不是本课题的主要部分，但也不能忽视的，本文在界面设计上花费了大量的时间和精力。当然，在老师的指导下， 在老师身上学到了不少好的习惯， 对论文的编写和系统的实现有很大的帮助。

我所做的软件的根本目的在于让老师能够准确的了解到自己教学的成果， 对自己以后在教学中需要做什么， 如何做有一定的了解和想法。 开发过程中， 考虑到用户的实际需要，对系统进行了最大可能的简化，使得用户能更方便的操作，快捷的查询，对结果一目了然 。

尽管在设计过程中遇到了很多问题， 但经过自己不懈的努力和老师的指导最终还是完成了系统。 或许所做的系统不是很好， 却是我自己一个人所完成的， 里边有自己的努力，有自己的心血， 系统最后做出来还是挺开心的。 如果说在实习当中意识到了团队合作的重要性，那么通过本次设计的实现让我意识到了个人实力和独立完成某个任务的必要性。本次论文和设计让我个人的写作能力，动手能力，阅读能力等有了很大的提升，对我未来的生活和学习有很大的帮助。

# 致　谢

伴随着毕业设计的完成，　大学的四年时光也即将划上一个圆满的句号。　在这短暂的毕业设计期间，有付出有回报，有欣喜有感动，太多的事，太多的人须要我铭记在心，太多的汗水，　太多的欢笑须要我在未来的日子里细细品尝。　看着眼前的作品，　那些日子我们一起走过的画面时时闪现，感谢有你一直陪在我的身边。

首先我要感谢我的导师，　正是有了他的悉心指导和谨慎负责的态度和工作作风才让我在毕业设计中少走了许多弯路，克服了众多困难，更好地取得成功，同时，他们高深的学术造诣和勤于钻研，　戒骄戒躁，　一丝不苟的作风也值得我在未来的学习和工作中继承和发扬，　正是他们的付出和谆谆教导为我的毕业设计奠定了坚定的基础，　作出了最有力的支撑，　我的每次小小的进步都离不开他们细心耐烦的讲解和帮助，　因此，　在这里我以感恩的心表达我最崇高的敬意。

其次感谢学长们对我的莫大帮助，不管多小的问题他们都会以百倍的热情为我解答，在程序的调试过程中，他们经常手把手地帮扶，并毫无保留地传授相关经验，使我获益匪浅，也使我更快地进入角色，完成相关工作，虽然时间短暂，但足以让我铭记于心，在此，向他们表达诚挚的感谢。

第三我要感谢这四年来给我带过课的所有老师，　不是你们的细心教导，　我就不会有太多的积累来完成本次论文和设计，　在这里我向你们表示深深地感谢！　还有感谢学长们对我的莫大帮助，　不管多小的问题他们都会以百倍的热情为我解答，　在程序的编写和调试过程中，他们经常手把手地帮扶，并毫无保留地传授相关经验，使我获益匪浅，也使我更快地进入角色，完成相关工作，虽然时间短暂，但足以让我铭记于心，在此，向他们表达诚挚的感谢。我要感谢我的父母，正是父母的辛勤养育，背后的默默支持，关心和呵护让我即使在心灰意冷时也会拥有最温暖的心灵港湾，　在这里我想以最感恩的心表达此时此刻的心情，您们辛苦了！

最后感谢在百忙之后审阅本文的各位专家、教授谨向您们表示最衷心的敬意！

# 参考文献


[1] 布兰切特 . C++ GUI Qt 4编程 [M]. 电子工业出版社 , 2013.

[2]( 美)艾朱斯特 (Ezust, A. ), (美)艾朱斯特 (Ezust,等 . C++ Qt设计模式 [M]. 电子工业出版社 , 2012.

[3] 陈琦 . QT的编程技术及应用 [J]. 科技信息 , 2008(33).

[4] 吴迪 . 零基础学 Qt4编程 [M]. 北京航空航天大学出版社 , 2010.

[5] 索兰 . 24小时学通 Qt编程 [M]. 人民邮电出版社 , 2000.

[6] 刘晓立 , 赵俊逸 . 基于Qt的音乐播放器 [J]. 软件导刊 , 2015, 14(10):112-114.

[7] 杨芙清 . 软件工程技术发展思索 [J]. 软件学报 , 2005, 16(1):1-7.

[8] 焦正才 , 樊文侠 . 基于 Qt/Embedded的MP3 音乐播放器的设计与实现 [J]. 电子设计工程, 2012, 20(7):148-150.

[9] 李攀 , 田江丽 . 一种基于 QT/E 的嵌入式车载播放系统的设计方案 [J]. 济源职业技术学院学报 , 2012(4):24-28.

[10] 谭大鹏 , 李培玉 , 潘晓弘 . 基于Qt/E的嵌入式工业监测轻型图形用户界面构件库开发[J]. 计算机集成制造系统 , 2009, 15(2):399-405.

[11]刘汇丹 , 芮建武 , 姚延栋 ,等 . 基于 Qt的国际化图形用户界面设计与实现 [J]. 中文信息学报 , 2006, 20(4):94-99.

[12]倪红波 , 周兴社 , 谷建华 . 基于 QT/E 的嵌入式图形支持系统 [J]. 计算机工程 , 2007, 33(20):256-258.

[13]朱吉佳 , 蔡家麟 . 基于 Qt的业务监控系统界面设计与实现 [J]. 计算机技术与发展 , 2008, 18(3):236-239.

[14] 彭均键 , 史步海 , 刘洋 . 基于 Qt的嵌入式 GUI 开发平台的搭建 [J]. 微型电脑应用 , 2010, 26(2):40-42.

[15] 王建民 , 张宏壮 . 基于 Qt的嵌入式媒体播放器系统的设计 [J]. 微计算机信息 , 2008, 24(20):64-66.

[16] 李艳民 . 基于 Qt跨平台的人机交互界面的研究和应用 [D]. 重庆大学 , 2007.

[17] 黄艳芳 . 基于 Qt4的图形用户界面程序设计与游戏开发 [J]. 电子设计工程 , 2011,

19(17):49-53.

[18] 李强 . DJ论道 [M]. 福建人民出版社 , 2005.

[19] 张栋 .LINUX 服务器搭建实战详解 . 北京 :电子工业出版社出版 ,2010.

[20] 孙琼 . 嵌入式 Linux 应用程序开发详解 [M]. 人民邮电出版社 , 2006.

[21] 韩少云 . 基于嵌入式 Linux 的Qt图形程序实战开发 [M]. 北京航空航天大学出版社 , 2012.

[22]Meyers S. More Effective C++: 35 New Ways to Improve Your Programs and Designs[J]. Pearson Schweiz Ag, 1996.

[23]Weiskamp K, Flamig B. The complete C++ primer[M]. AP Professional, 1989.

[24] 霍亚飞 . Qt Creator快速入门 [M]. 北京航空航天大学出版社 , 2012.

[25] 霍亚飞 . Qt及Qt Quick开发实战精解 [M]. 北京航空航天大学出版社 , 2012.

[26] Blum R. Linux Command Line and Shell Scripting Bible[J]. Apress, 2011.

[27]《24小时学通 QT编程》 著：Daiel Solin 译：袁鹏飞 人民邮电出版社

[28] KDE2/QT 编程宝典 电子工业出版社

[29] ARM 嵌入式 linux 系统开发 电子工业出版社

[30］W.Richard Stevenson.StephenA.Rago.UNIX环境高级编程［ M］.人民邮电出版社, 2006.

[31］StanleyB.Lippman And JOSEE Lajoie And BarbaraE.Moo. C++ Pramer Plus［M］.第五版 .人民邮电出版社

[32] Jasmin Blanchette And Mark Summer field. C++ GUI Program –mingwith QT4 ［M］.第二版 .电子工业出版社, 2008.

[33］蔡志明，卢传富等 .精通 Qt4编程［ M］.电子工业出版社, 2008

# 附录

软件核心部分代码

//main.cpp

```
#include "MainInterface.h"
#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainInterface w;
    w.show();

    return a.exec();
}
```

//LrcInterface.cpp

```
LrcInterface::LrcInterface(QWidget *parent) :
    QLabel(parent)
{
    this->setWindowFlags(Qt::SubWindow |Qt::FramelessWindowHint|
Qt::WindowStaysOnTopHint);
    this->resize(1024,60);
    this->setText(tr("Music ..."));
    this->setAttribute(Qt::WA_TranslucentBackground);

    this->setCursor(Qt::OpenHandCursor);

    exit = new QAction(tr(" 隐藏 (&D)"),this);
    connect(exit,SIGNAL(triggered()),this,SLOT(close()));

    timer = new QTimer(this);
    connect(timer,SIGNAL(timeout()),this,SLOT(timeout()));

    this->move(400,700);

    lrcWidth = 0;
}

void LrcInterface::mousePressEvent(QMouseEvent *event)
{
```

```
if(event->button() == Qt::LeftButton)
{
	dragPosition = event->globalPos() - frameGeometry().topLeft();
	event->accept();
}
else if(event->button() == Qt::MidButton)
	close();
	QLabel::mousePressEvent(event);
}

void LrcInterface::mouseMoveEvent(QMouseEvent *e)
{
	if(e->buttons() & Qt::LeftButton)
	{
		move(e->globalPos() - dragPosition);
		e->accept();
	}
	QLabel::mouseMoveEvent(e);
}

void LrcInterface::contextMenuEvent(QContextMenuEvent *ev)
{
	QMenu menu;
	menu.addAction(exit);
	menu.exec(ev->globalPos());
	QLabel::contextMenuEvent(ev);
}

void LrcInterface::paintEvent(QPaintEvent *)
{
	QPainter painter(this);
	painter.setRenderHints(QPainter::Antialiasing | QPainter::TextAntialiasing);

QFont font(tr("Times New Roman"),30,QFont::Bold);
painter.setFont(font);
QLinearGradient lg(0,20,0,50);
lg.setColorAt(0,QColor(0,170,255,255));
lg.setColorAt(0.2,QColor(61,214,191,250));
lg.setColorAt(0.5,QColor(85,255,255,255));
lg.setColorAt(0.8,QColor(61,214,191,250));
lg.setColorAt(1,QColor(0,170,255,255));
painter.setBrush(lg);
```

```
painter.setPen(Qt::NoPen);
QPainterPath textPath;
textPath.addText(0,50,font,text());
painter.drawPath(textPath);

length = textPath.currentPosition().x();
        painter.setPen(Qt::yellow);
        painter.drawText(0,14,lrcWidth,50,Qt::AlignLeft,text());

}

void LrcInterface::timeout()
{
        lrcWidth += length/85;
        update();
}

void LrcInterface::setLrcWidth()
{
        lrcWidth = 0;
}
// MainInterface.cpp
MainInterface::MainInterface(QWidget *parent) :
        QWidget(parent),ui(new Ui::MainInterface),isTouched(false)
{
        ui->setupUi(this);

        this->setAcceptDrops(true);
        this->setFixedSize(300,150);
        this->setWindowTitle(tr(" 聆听， 播放器 "));
        readSettings();

        m_text = new QTextEdit(this);
        m_text->hide();
        ui->textEdit-> setVisible(false);

        m_playList = new MusicList(this);
        m_playList->setFixedSize(300,320);
        m_playList->ui->tableWidget->setColumnWidth(0,300);
        m_playList->ui->tableWidget->setColumnWidth(1,0);
```

```
connect(m_playList->ui->tableWidget,SIGNAL(cellDoubleClicked(int,int)),SLOT(tableWid
get_cellDoubleClicked(int)));
        connect(m_playList,SIGNAL(listClose()),ui->toolButton_List,SLOT(toggle()));

        m_playList->installEventFilter(this);
        m_playList->ui->tableWidget->installEventFilter(this);


        m_volSlider = new QSlider(this);
        m_volSlider->move(190,80);
        m_volSlider->resize(50,20);
        m_volSlider->setStyleSheet("background-color:rgb(255,255,255,100)");
        m_volSlider->setFixedWidth(100);
        m_volSlider->setOrientation(Qt::Horizontal);
        m_volSlider->setRange(0,100);

        m_seekSlider = new QSlider(this);
        m_seekSlider->move(10,35);
        m_seekSlider->resize(170,20);
        m_seekSlider->setStyleSheet("background-color:rgb(255,255,255,100)");
        m_seekSlider->setOrientation(Qt::Horizontal);
        connect(m_seekSlider,SIGNAL(sliderMoved(int)),SLOT(setPosition(int)));

        m_lrc = new LrcInterface(NULL);
        QPalette palette = ui->label_lrc->palette();
        palette.setColor(QPalette::WindowText,Qt::darkBlue);
        ui->label_lrc->setPalette(palette);


        m_audio = new QMediaPlayer(this);
        m_audioList =new QMediaPlaylist(this);
        m_audio->setPlaylist(m_audioList);
        m_audio->setNotifyInterval(10);
        m_volSlider->setValue(m_audio->volume());

        connect(m_audio, SIGNAL(positionChanged(qint64)),
SLOT(positionChanged(qint64)));

connect(m_audio,SIGNAL(stateChanged(QMediaPlayer::State)),SLOT(audioStateChanged(
QMediaPlayer::State)));
```

```
    connect(m_audioList, SIGNAL(currentIndexChanged(int)),
SLOT(audiolistPositionChanged(int)));

    connect(m_volSlider,SIGNAL(valueChanged(int)),m_audio,SLOT(setVolume(int)));
    connect(m_audio,SIGNAL(volumeChanged(int)),m_volSlider,SLOT(setValue(int)));

    creatActions();
}

MainInterface::~MainInterface()
{
    delete ui;
}

void MainInterface::audioStateChanged(QMediaPlayer::State state)
{
    switch(state)
    {
        case QMediaPlayer::PlayingState:
            {
                m_lrc->setText("Music ...");
                ui->label_lrc->setText("Music ...");
                loadCurrentLrc();
                play->setIcon(*iconpause);
                ui->toolButton_playpause->setIcon(*iconpause);
                ui->toolButton_playpause->setToolTip(tr("暂停"));
                play->setText(tr("暂停"));
                QString fileName = m_audio->currentMedia().canonicalUrl().toString();
                QString title = fileName.right(fileName.length() -
fileName.lastIndexOf('/') - 1);
                this->setWindowTitle(title);
                ui->label_palyname->setText(title);
                trayicon->setToolTip(tr("播放")+ title);
                break;
            }
        case QMediaPlayer::PausedState:
            {
                play->setIcon(*iconplay);
                ui->toolButton_playpause->setIcon(*iconplay);
                ui->toolButton_playpause->setToolTip(tr(""));
                ui->label_palyname->setText(tr("暂停播放"));
                play->setText(tr("播放"));
```

```
                trayicon->setToolTip(tr(" 暂停 "));
                m_lrc->timer->stop();

                break;
        }
case QMediaPlayer::StoppedState:
        {
                ui->toolButton_playpause->setIcon(*iconplay);
                play->setText(tr("播放 "));
                play->setIcon(*iconplay);
                this->setWindowTitle(tr("Music .."));
                ui->label_palyname->setText(tr("Music .."));
                trayicon->setToolTip(tr("stop"));
                ui->label_time->setText("00:00 / 00:00");
                m_lrc->setText("Music ...");
                ui->label_lrc->setText("Music ...");
                m_lrc->timer->stop();
                break;
        }
                default:break;
        }
}

void MainInterface::setPosition(int value)
{
        m_audio->setPosition(qint64(value));
}


void MainInterface::positionChanged(qint64 time)
{
        qint64 temp = m_audio->duration();
        QTime totalTime(0,(temp / 60000) % 60,(temp / 1000) % 60,time %1000);
        QTime curTime(0,(time / 60000) % 60,(time / 1000) % 60,time %1000);
        ui->label_time->setText(tr("%1
/ %2").arg(curTime.toString("mm:ss")).arg(totalTime.toString("mm:ss")));
        ui->label_time->update();

        m_seekSlider->setRange(0, temp);
        m_seekSlider->setValue(time);
```

```
        if(ui->textEdit->find(curTime.toString("mm:ss.zzz").left(7)))
        {
                QString str =
ui->textEdit->textCursor().block().text().replace(QRegExp("\\[\\d{2}:\\d{2}\\.\\d{2}\\]"),"");
                ui->label_lrc->setText(str);
                m_lrc->setText(str);
                m_lrc->setLrcWidth();

                QTime tt = curTime;
                int b = 1,c= 0;
                m_text->setText(ui->textEdit->document()->toPlainText());
                bool over = ui->textEdit->textCursor().block().next().text().isEmpty();
                while(!over &&!m_text->find(tt.addMSecs(b*100).toString("mm:ss.zzz").left(7)))
                {
                        b++;    c++;
                }
                while(over
                && !m_text->find(tt.addMSecs(b*100).toString("mm:ss.zzz").left(7),QTextDocument::Find
                Backward))
                        {
                                b++;    c++;    break;
                        }
                        m_lrc->timer->start(c);
        }
        else
if(ui->textEdit->find(curTime.toString("mm:ss.zzz").left(7),QTextDocument::FindBackward
))
        {
                QString str =
ui->textEdit->textCursor().block().text().replace(QRegExp("\\[\\d{2}:\\d{2}\\.\\d{2}\\]"),"");
                ui->label_lrc->setText(str);
                m_lrc->setText(str);
                m_lrc->setLrcWidth();

                QTime tt = curTime;
                int b = 1;
                int c= 0;
                m_text->setText(ui->textEdit->document()->toPlainText());
```

```
bool over = ui->textEdit->textCursor().block().next().text().isEmpty();
while(!over && !m_text->find(tt.addMSecs(b*100).toString("mm:ss.zzz").left(7)))
{
        b++;      c++;
}
while(over
&& !m_text->find(tt.addMSecs(b*100).toString("mm:ss.zzz").left(7),QTextDocument::Find
Backward))
                {
                        b++;    c++;      break;
                }
                m_lrc->timer->start(c);
        }
}

void MainInterface::audiolistPositionChanged(int index)
{
        m_playList->ui->tableWidget->selectRow(index);
}

void MainInterface::creatActions()
{
        QIcon iconremove(":/images/remove.png");
        QIcon iconstop(":/images/gtk-media-stop.png");
        QIcon iconopen(":/images/gtk-open.png");
        QIcon iconsound(":/images/sound.png");
        QIcon iconexit(":/images/exit.png");
        iconpause = new QIcon(":/images/gtk-media-pause.png");
        iconplay = new QIcon(":/images/gtk-media-play-ltr.png");

        remove = new QAction(iconremove,tr("clear playlist"),this);

        connect(remove,SIGNAL(triggered()),this,SLOT(clearPlayList()));

        play = new QAction(*iconplay,tr("play"),this);
        connect(play,SIGNAL(triggered()),this,SLOT(on_toolButton_playpause_clicked()));

        stop = new QAction(iconstop,tr("stop"),this);
        connect(stop,SIGNAL(triggered()),m_audio,SLOT(stop()));

        open = new QAction(iconopen,tr("open file"),this);
        connect(open,SIGNAL(triggered()),this,SLOT(on_toolButton_open_clicked()));
```

```
sound = new QAction(iconsound,tr("no voice"),this);
sound->setCheckable(true);
connect(sound,SIGNAL(triggered(bool)),m_audio,SLOT(setMuted(bool)));

exit = new QAction(iconexit,tr("quit"),this);
connect(exit,SIGNAL(triggered()),this,SLOT(close()));


trayicon = new QSystemTrayIcon(this);
QIcon ico(":/images/VMP2.png");
trayiconMenu = new QMenu(this);
trayiconMenu->addAction(play);
trayiconMenu->addAction(stop);
trayiconMenu->addAction(open);
trayiconMenu->addSeparator();
trayiconMenu->addAction(exit);

trayicon->setIcon(ico);
trayicon->setToolTip(tr("playinging"));
        trayicon->setContextMenu(trayiconMenu);

        trayicon->show();

trayicon->showMessage(tr("mp3"),tr("www.yafeilinux.com"),QSystemTrayIcon::Informatio
n,10000);


connect(trayicon,SIGNAL(activated(QSystemTrayIcon::ActivationReason)),SLOT(iconActi
vated(QSystemTrayIcon::ActivationReason)));//图标被激活时
}

void MainInterface::loadCurrentLrc()
{
        QString mp3Name = m_audio->currentMedia().canonicalUrl().toString();
        if(mp3Name.contains(QRegExp("file:///")))
                mp3Name= mp3Name.remove(0,8);
        QString lrcName = mp3Name.remove(mp3Name.right(3)) + "lrc";
        QFile file(lrcName);
        if(file.exists())
        {
                file.open(QFile::ReadOnly | QFile::Text);
```

```
        QTextStream in(&file);
        ui->textEdit->setText(in.readAll());
}
else
{
        ui->textEdit->clear();
        ui->label_lrc->setText(tr("listen to your heart"));
        m_lrc->setText(tr("listen to your heart"));
    }

}

void MainInterface::iconActivated(QSystemTrayIcon::ActivationReason reason)
{
    switch(reason)
    {
    case QSystemTrayIcon::Trigger:
    case QSystemTrayIcon::DoubleClick:
        this->readSettings();
        this->setWindowState(Qt::WindowActive);
        this->show();
        if(ui->toolButton_List->isChecked())
        {
            m_playList->show();
            m_playList->listReadSettings();
        }
        break;
    case QSystemTrayIcon::MiddleClick:
        trayicon->showMessage(tr("mp3"),tr("!!"),QSystemTrayIcon::Information,10000);
        break;
    default:
        ;
    }
}


void MainInterface::clearPlayList()
{
    m_playList->clearList();
    m_audio->stop();
    m_audioList->clear();
}
```

```
bool MainInterface::eventFilter(QObject *object, QEvent *event)
{
	if(object == m_playList->ui->tableWidget)
	{
		if(event->type() == QEvent::ContextMenu)
		{
			QContextMenuEvent *contextevent = static_cast<QContextMenuEvent
*>(event);
			QMenu menu(this);
			menu.addAction(remove);
			menu.exec(contextevent->globalPos());
			return true;
		}
		else if(event->type() == QEvent::DragEnter)
		{
			QDragEnterEvent *dragEnterevent = static_cast<QDragEnterEvent *>(event);
			if(dragEnterevent->mimeData()->hasFormat("text/uri-list"))
				dragEnterevent->acceptProposedAction();
			return true;
		}
		else if(event->type() == QEvent::Drop)
		{
			QDropEvent *dropevent = static_cast<QDropEvent *>(event);
			dropEvent(dropevent);
			return true;
		}
		else return false;
	}

	return QWidget::eventFilter(object,event);
}

void MainInterface::dragEnterEvent(QDragEnterEvent *e)
{
	if(e->mimeData()->hasFormat("text/uri-list"))
		e->acceptProposedAction();
}

void MainInterface::dropEvent(QDropEvent *event)
{
	QList<QUrl> urls = event->mimeData()->urls();
```

```
if (urls.isEmpty())return;
QString fileName = urls.first().toLocalFile();
if (fileName.isEmpty()) return;
int index =m_playList->ui->tableWidget->rowCount();
m_audio->stop();
for(int i=0; i<urls.length(); i++)
        {
            QMediaContent con(urls.at(i).toString());
            m_audioList->addMedia(con);
            QString    fileName =urls.at(i).toString();
            QString title = fileName.right(fileName.length() - fileName.lastIndexOf('/') - 1);
            m_playList->ui->tableWidget->insertRow(index+i);
            QTableWidgetItem *titleItem = new QTableWidgetItem(title);
            m_playList->ui->tableWidget->setItem(index+i,0,titleItem);
        }
        m_audioList->setCurrentIndex(index);
        m_audio->play();
}

void MainInterface::closeEvent(QCloseEvent *e)
{
        writeSettings();
        m_playList->listWriteSettings();
        if(this->isVisible())
        {
                this->hide();
                m_playList->hide();
                trayicon->showMessage(tr("player"),tr("mixed to
tupan"),QSystemTrayIcon::Information,15000);
                e->ignore();
        }
        else
                qApp->quit();
}

void MainInterface::wheelEvent(QWheelEvent *wheelevent)
{
        if(wheelevent->delta() > 0 )
        {
                int newvolume = m_audio->volume()+1;
                if(newvolume >= 100)    newvolume = 100;
```

```
        m_audio->setVolume(newvolume);
}
else
{
        int newvolume = m_audio->volume()-1;
        if(newvolume <= 0)     newvolume = 0;
                m_audio->setVolume(newvolume);
        }
}

void MainInterface::contextMenuEvent(QContextMenuEvent *e)
{
        QMenu menu(this);
        menu.addAction(play);
        menu.addAction(stop);
        menu.addAction(open);
        menu.addSeparator();
        menu.addAction(sound);
        menu.addSeparator();
        menu.addAction(exit);
        menu.exec(e->globalPos());
}

void MainInterface::moveEvent(QMoveEvent *e)\
{
        if(isTouched)
        {
                QPoint p = e->pos() - e->oldPos();
                m_playList->move(m_playList->pos() + p);
        }
        else
        {
                isTouched = false;
        if(qAbs(m_playList->y()- this->y() - this->frameGeometry().height()) < 20)
        {
                isTouched = true;
                this->move(this->x(),m_playList->y()+this->frameGeometry().height());
        }
        if(qAbs(this->x() - m_playList->x() - m_playList->frameGeometry().width()) < 20)
        {
                isTouched = true;
```

```
this->move(m_playList->x()+m_playList->frameGeometry().width(),this->y());
            }
            if(qAbs(this->x() + this->frameGeometry().width() - m_playList->x()) < 20)
            {
                isTouched = true;
                this->move(m_playList->x()-this->frameGeometry().width(),this->y());
            }
        }
        QWidget::moveEvent(e);
}

void MainInterface::tableWidget_cellDoubleClicked(int row)
{
        m_audio->stop();
        if (row >= m_audioList->mediaCount())       return;
        m_audio->setMedia(m_audioList->media(row));
        m_audio->play();
}

void MainInterface::on_toolButton_lrcD_clicked(bool checked)
{
        checked? m_lrc->show(): m_lrc->hide();
}

void MainInterface::on_toolButton_next_clicked()
{
        int index = m_audioList->nextIndex();
        if(index < m_audioList->mediaCount())
        {
            m_audio->stop();
            m_audio->setMedia(m_audioList->media(index));
            m_audio->play();
            m_audioList->setCurrentIndex(index);
        }
}

void MainInterface::on_toolButton_previous_clicked()
{
        int index = m_audioList->previousIndex();
        if(index >= 0)
        {
```

```
            m_audio->stop();
            m_audio->setMedia(m_audioList->media(index));
            m_audio->play();
            m_audioList->setCurrentIndex(index);
        }
}

void MainInterface::on_toolButton_stop_clicked()
{
        m_audio->stop();
}

void MainInterface::on_toolButton_playpause_clicked()
{
        if(m_audio->state() == QMediaPlayer::PlayingState)
                m_audio->pause();
        else
                m_audio->play();
}
//打开
void MainInterface::on_toolButton_open_clicked()
{
        QStringList urls = QFileDialog::getOpenFileNames(this,tr("open the mp3 file"),
QStandardPaths::writableLocation(QStandardPaths::MusicLocation));
        if(urls.isEmpty()) return;
        int index = m_playList->ui->tableWidget->rowCount();
        for(int i=0; i<urls.length(); i++)
        {
            QMediaContent con(urls.at(i).trimmed());
            m_audioList->addMedia(con);
            QString    fileName =urls.at(i).trimmed();
            QString title = fileName.right(fileName.length() - fileName.lastIndexOf('/') - 1);
            m_playList->ui->tableWidget->insertRow(index+i);
            QTableWidgetItem *titleItem = new QTableWidgetItem(title);
            m_playList->ui->tableWidget->setItem(index+i,0,titleItem);
        }

        m_audioList->setCurrentIndex(index);
        m_audio->play();

}
```

```
void MainInterface::on_toolButton_List_clicked(bool checked)
{
	checked? m_playList->show(): m_playList->hide();
}

void MainInterface::readSettings()
{
	QSettings settings("test", "Main");
	QPoint pos = settings.value("pos", QPoint(400,200)).toPoint();
	this->move(pos);
}

void MainInterface::writeSettings()
{
	QSettings settings("test","Main");
	settings.setValue("pos", pos());
}
```

//MusicList.cpp

```
#include "MusicList.h"
#include <QDebug>

MusicList::MusicList(QWidget *parent) :
	QWidget(parent),ui(new Ui::MusicList)
{
	ui->setupUi(this);
	listReadSettings();
	ui->tableWidget->setAcceptDrops(true);
	this->setWindowFlags(Qt::Tool    );
}

MusicList::~ MusicList()
{
	delete ui;
}

void MusicList:: changeEvent(QEvent *e)
{
	QWidget::changeEvent(e);
	switch (e->type()) {
	case QEvent::LanguageChange:
		ui->retranslateUi(this);
```

```
            break;
        default:
            break;
        }
}

void MusicList:: closeEvent(QCloseEvent *e)
{
        emit listClose();
        listWriteSettings();
        e->accept();
}

void MusicList::listReadSettings()
{
        QSettings settings("test", "List");
        move(settings.value("pos", QPoint(this->x(),this->y()+150)).toPoint());
}
  void MusicList::clearList()
  {
          int count = ui->tableWidget->rowCount();
          for(int i=0;i<count;i++)
                  ui->tableWidget->removeRow(0);
          ui->tableWidget->close();
          ui->tableWidget->show();
   }

void MusicList::listWriteSettings()
{
        QSettings settings("test","List");
        settings.setValue("pos", pos());
}

void MusicList:: moveEvent(QMoveEvent *e)
{
        if(qAbs((this->y()- this->parentWidget()->y() -
this->parentWidget()->frameGeometry().height())) < 20)
        {
                this->move(this->x(),this->parentWidget()->y() +
this->parentWidget()->frameGeometry().height());
        }
        if(qAbs(this->x()+this->frameGeometry().width() - this->parentWidget()->x()) < 20)
```

```
        {
            this->move(this->parentWidget()->x()-this->frameGeometry().width(),this->y());
        }

if(qAbs(this->x()-this->parentWidget()->frameGeometry().width()-this->parentWidget()->x(
)) < 20)
        {

this->move(this->parentWidget()->x()+this->parentWidget()->frameGeometry().width(),this-
>y());
        }
        QWidget::moveEvent(e);
}
```