**基于Qt技术的音视频播放器的设计与实现**

计算机与信息科学学院 软件工程专业 2019级 田润

**摘 要：**随着时间的推移,电脑已经开始作为我们日常生活的主力军,作用也日益巨大。更多的人利用通过网络中的视频来了解周围突发的事物来改变自身的日常生活方式,而视频播放器的类型也越来越多,功能越来越繁杂。 本论文基于Qt框架,设计并实现了一款音视频播放器。该播放器具有良好的用户界面和稳定的播放性能,支持多种音视频格式的播放和快速切换。在实现过程中,采用了Qt的多媒体模块和OpenGL技术,实现了视频的硬件加速和高效渲染。同时,通过对Qt的多线程机制的深入理解和应用,实现了音视频的异步加载和播放,提高了播放器的响应速度和稳定性。在测试中,该播放器表现出了良好的性能和用户体验,具有一定的实用价值。

关键词:Qt;C++;视频;播放器;Gui设计。

**Abstract：**As time goes on, computers have become an increasingly important part of our daily lives. More and more people are using videos on the internet to change their daily life by understanding what is happening around them, and there are more and more types of video players with more and more functions. This thesis designs and implements an audio and video player based on the Qt framework. The player has a good user interface and stable performance, and supports a variety of audio and video formats and fast switching. In the process of implementation, the multimedia module of Qt and OpenGL technology are used to achieve hardware acceleration and efficient rendering of video. At the same time, through the in-depth understanding and application of Qt's multi-threading mechanism, the asynchronous loading and playback of audio and video is achieved, which improves the response speed and stability of the player. In tests, the player has shown good performance and user experience, and has some practical value.

**Key words：Q**t;c++;Video;Player;Gui Design.

1.引言

如今,由于科学技术的高速发展,电脑和网络也开始在人类的日常生活中横行。科学技术的提高也伴随着生产技术水平的提升,人们对计算机软件的需求也愈来愈大。而现在人们周围有不少播放器,比如:电视、电脑、手机、MP3、MP4、MP5等电子产品。其软件功能通常都比较为简单,对文件内容格式的功能却很少,实际应用起来也存在着许多麻烦。为克服市场上常见产品的某些不足,我们只能采用Windows操作系统的Qt设计软件来完成一个多媒体播放器。首先,能够通过Qt的C++图形用户界面程序来完成多媒体播放器的外观和外形,从而初步完成了多媒体播放器的基本外观和框架。其次,通过Qt软件中提供的种类和槽能够完成播放列表和文件的功能,以及多种格式功能[1]。

本论文基于Qt框架,设计并实现了一款音视频播放器。该播放器具有良好的用户界面和稳定的播放性能,支持多种音视频格式的播放和快速切换。在实现过程中,采用了Qt的多媒体模块和OpenGL技术,实现了视频的硬件加速和高效渲染。同时,通过对Qt的多线程机制的深入理解和应用,实现了音视频的异步加载和播放,提高了播放器的响应速度和稳定性。在测试中,该播放器表现出了良好的性能和用户体验,具有一定的实用价值。

本设计的最终目标是实现一种简单容易用的视频播放器,能够用来回放、暂停、快进和快退出等一些视频内容。多格式的功能也将被实现。由于这个播放器是用Qt语言写成的,所以Qt出色的可移植性也使得人们能够将其迁移到我们所需要的装置上,从而扩大了这个播放器的功效[2]。如果可以的话,我期待人们有机会将该程序迁移到开发板上,以方便人们随时随地应用。

1. **1项目概述**

该项目是一个基于Qt框架开发的音频和视频播放器。它的目的是提供一个易于使用、功能丰富的播放器,支持播放各种音频和视频格式,以及常见的播放控制。该播放器有一个现代化的UI设计,允许用户通过简单的操作来播放、暂停、快进、快退和调整音频和视频的音量。

这个项目的主要特点包括:

支援各种声音和录像形式:该播送器支援常用的声音和录像形式,如MP4、AVI、MKV、FLV等。用户可以轻松播放各种格式的音频和视频文件。

易于应用的放映功能:该放映器具有常用的放映功能,如放映、暂停、快进、快退、声音调节等。用户可以通过简单的操作实现这些功能。

现代化的用户界面设计:这款播放器采用现代UI设计,界面简洁美观,用户可以轻松找到自己需要的功能。

支持全屏播放:该播放器支持全屏播放,用户可以在全屏模式下享受更好的观看体验。

多语言支持:本播放器支持多国语言,用户可以选择自己熟悉的语言进行操作。

总之,这个项目是一种功能丰富且容易应用的音乐与视频播放器,适于广大用户使用。

**1.2研究背景**

随着网络时代的来临,人类获得消息的方法也从报刊、图书、口口相传,逐步升级为收音机、电视、电脑、手机。从文本到音频再到视频,技术的发展往往伴随着生活品质的提升,音乐与视频播放器也开始成为了人类生活中不可或缺的重要组成部分。过去,人们主要使用传统的媒体播放器,如CD和DVD,但随着互联网的普及和数字媒体的兴起,人们开始更多地使用电脑、手机和其他设备来播放音频和视频。因此,开发一个高效、稳定、易用的音视频播放器已成为一个重要的研究方向。

Qt是一种跨网络平台的C++技术应用开发框架,其提供了大量的绘图用户界面、网络通信、数据库管理系统等功能,被广泛于各类应用程式的开发[3]。在音视频播放器的开发中,Qt可以提供良好的用户界面设计、多媒体框架、网络通信等功能,使开发者更容易实现音视频播放器的各种功能。Qt是面向对象的架构,可以使用特定的代码产生扩展包括一个宏,用于扩充,并可以对组件编程。拥有出色的跨平台稳定性,丰富多彩的API,并且完善的封装机制使Qt的复用度很高。提供了二D/三D图像渲染,并使用OpenGL,也就是由于上述优势使Qt受到了广泛的使用。这也就是由于Qt优秀的可移植性,我决定采用Qt来编写这款应用软件,从而实现了在除了Windows之外的所有操作系统下正常工作的所有功能。

因此,对基于Qt的音视频播放器的研究具有重要意义。通过对Qt多媒体框架的深入研究和应用,可以开发出高效、稳定、易用的音视频播放器,满足人们在日常生活中对音视频播放的需求。同时,对基于Qt的音视频播放器的研究也可以为其他应用的开发提供参考和借鉴,促进数字媒体技术的发展。

**1.3论文的主要内容和设计目标**

本论文设计并实现了一个基于Qt平台的音频和视频播放器。该播放器实现了基本的音视频格式解码和播放功能,包括支持常见的音视频格式,如mp4、avi、mkv等。该播放器还提供了一些扩展功能,如支持选择不同的音频和视频轨道,支持音频和视频调整(如音量、亮度等)和全屏播放。

在实现过程中,我们主要使用了Qt的多媒体框架QMediaPlayer,并定制了QVideoWidget来实现播放器的UI定制[4]。为增强使用感受,在回放页面给出了详尽的操作按键,包括播放/停止、上一帧/下一帧、快进/快退等。

设计目标:

（1）这个音频/视频播放器的设计目标包括以下内容:

（2）实现常见音频和视频格式的解码和播放功能,为用户提供良好的播放体验。

（3）实现音频和视频的多轨选择和调整功能,以改善用户的体验。

（4）提供详细的控制按钮和UI定制功能,让用户更容易控制播放器。

（5）最大限度地提高播放器的效率和速度,避免滞后和延迟。

（6）实现全屏播放功能,使用户能够享受到更加身临其境的播放体验。

总之,这个音频/视频播放器的目标是提供一个稳定的、功能丰富的、易于使用的音频/视频播放器,让用户更方便地播放各种类型的音频/视频文件,获得更好的播放体验。

**2 相关技术介绍**

2.1 Qt音频和视频技术原理

Qt音频技术提供了对音频设备的访问、音频数据的处理、音频播放和录制等功能。它支持多种音频格式，包括WAV、MP3、AAC等，并且可以进行音频数据的混音、降噪、回声消除等处理。Qt音频模块还提供了多种音频播放和录制接口，包括QAudioInput、QAudioOutput、QMediaPlayer等[5]。

Qt视频技术提供了对视频设备的访问、视频数据的处理、视频播放和录制等功能。它支持多种视频格式，包括AVI、MP4、MKV等，并且可以进行视频数据的解码、编码、裁剪、合并等处理。Qt视频模块还提供了多种视频播放和录制接口，包括QCamera、QVideoWidget、QVideoEncoder等。

工作原理环绕着主从媒介流有中心,主媒介流为视讯流,从媒介流为声音流。同时视讯信息的解码速率维持恒定,可以根据操作系统时刻设定实际的时刻[6]。为能使音视频同时,还需要调节声音的传输速率。可以根据选定一条线性递增的本地操作系统时刻为基准时间(LSCR)。作用在视频解码器和声音解码器中。其次,根据各帧的数值比对自己时钟上的参考值。亦即在制作输出数据流时会,在各个数据库块上标明起始日期和终止时间。在放映这些数据分析时,则按照数据块上的日期来放映。

因为在广播过程中,只是针对有早到和晚到等事务的区块进行管理,而这种管理方式有着很大的不同,是远远不够的。所以通过合理的设计调整了多媒体播放器的功能,在设计时引入了反馈机制,根据对当前广播内容的播放情况来进行反馈机制。若声音信息滞后,则反馈信息提高声音的编码速率。若有视频信号滞后则反馈通知声音信息放慢编码速率来进行视频信号的同步。如果有任何一方滞后过多则直接切换至下一帧执行。

在分离器的分离下,将数据流划分成了声音信息和影像数据。采用正确的解码器解码后,再经过比对当前芯片中的系统时刻,在得到正确的显示时刻后进行比对。如果二者为相同时输出,不相同时产频信县进行竺待或猕帧外理,百到外理后达到同步十会输出。

2.2 多媒体框架技术

多媒体框架是一种软件框架，用于处理和管理多媒体数据，包括音频、视频、图像等。它提供了一组API和工具，可以帮助开发人员处理、编码、解码、播放和编辑多媒体数据。

FFmpeg是一个开源的跨平台音视频处理工具集，它可以对音视频进行格式转换、编解码、剪辑、截图、流媒体处理等操作。它既可以作为命令行工具使用，也可以通过各种编程语言的FFmpeg库进行集成开发。

FFmpeg最初是由Fabrice Bellard在2000年创建的，目前由Michael Niedermayer等人负责维护。它支持几乎所有的音视频格式和编解码器，并且可以在多种操作系统上运行，包括Windows、Linux、macOS等。此外，FFmpeg还支持硬件加速，可以提高音视频处理的速度[7]。

**3 需求分析**

用户分析也是研发下一个应用的一个关键环节,视频播放器也不例外。其实,需要因素在很大程度上决定着一份工作的好坏。供需分析主要是为了帮助人们明确的了解该做什么,怎样工作,需要哪些。

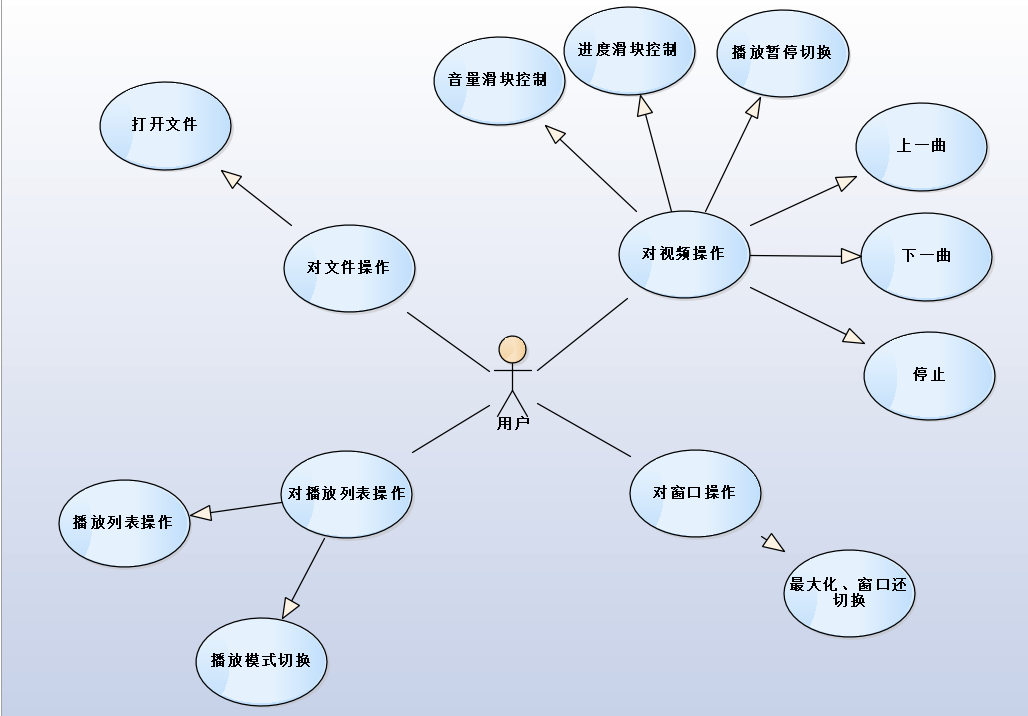


图3-1用例图

**3.1 技术可行性**

Qt作为一套跨平台的应用开发框架,已经被广泛应用于各种应用开发中。同时,Qt还提供了丰富的音视频相关的API,如QtMultimedia、QtAV[8]等,可以方便地实现音视频播放功能。因此,从技术角度来看,基于Qt的音视频播放器是可行的。

**3.2 市场需求的可行性**

随着互联网技术的不断发展,音视频播放器已经成为人们日常娱乐和学习的重要工具之一。同时,Qt作为一套跨平台的应用开发框架,也具有广泛的应用前景。因此,从市场需求的角度来看,基于Qt的音视频播放器是可行的。

**3.3 资源可行性**

本项目所需的主要资源包括人力资源和技术资源。人力资源包括开发人员和测试人员,技术资源包括开发工具和相关技术文献。在目前的环境下,这些资源是可以得到的。因此,从资源角度来看,基于Qt的音视频播放器是可行的。

FFmpeg的应用场景非常广泛，比如视频转码、压缩、剪辑、拼接、水印添加、字幕嵌入等。同时，它也是流媒体处理的重要工具，可以用来实现流媒体的推流、拉流、转码、分发等功能。许多流媒体服务商都使用了FFmpeg来实现自己的服务。

**3.4结论**

基于以上分析,基于Qt的音视频播放器是可行的。该项目具有良好的市场前景和技术基础,而且还可以得到足够的资源支持。因此,这个项目可以继续开展。

**4 多媒体播放器的设计**

**4.1软件架构设计**

该系统设计以现代C++程序的设计工作为背景,将Qt当作开发软件系统进行设计工作,并利用Qt内部所包装的库函数来设计系统功能,并以此来完成功能[9]。同时,利用介于Qt自身的技术优势(跨平台),大大提高了本操作系统的可移植性与扩展,从而提高了软件系统的实用性和适用性。

用户根据图形界面的基本窗口,可以按照在Qt软件里给出的图形库,来设置多媒体播放器的用户界面,软件的基本窗口,主要进行了启动、放映、暂停、停止等基本功能的按钮设置。同样可以连接槽函数,并透过对槽函数的创建来完成操作。同样,界面也有播放列表、加载文件等操作。

数据输入主要指用户通过点击或播放软件,或者播放软件对数据进行读取,通过特定的动作来反馈数据的动作,或者进行特定的操作。例如:选择播放内容、对按钮的响应、启动和禁用等功能。当用户界面上任何有对用户操作反馈的信息均为用户输入,并提供了适当的操作方法,软件才会做出相应的反馈。

封装类的调用,主要是用来完成软件系统构架和操作系统相互之间的连接功能。此播放器的应用中,其重点在于解决对底层函数调用接口和解码器的调整。采用这种类函数接口技术使得能够将与译码器相关的参数集中在一起,并利用类函数接口技术进行调整。由于客户确定变化的文本形式,由系统软件选择应答的解码器,如不能确定对应的译码器,则必须传递提示信息。由于所采用的封装类参数接口技术,从而使系统的可延续性获得了相应的提高。

视频编码的输出,是指解码器在完成解码后的音视频或文本数据流的输出。因为QT设计时对内核帧缓存区域是能够进行控制的。所以能够直接将编码好的信息,经由帧缓存区域发送到输出设备上去,从而实现了整个设计的解码输出效果。

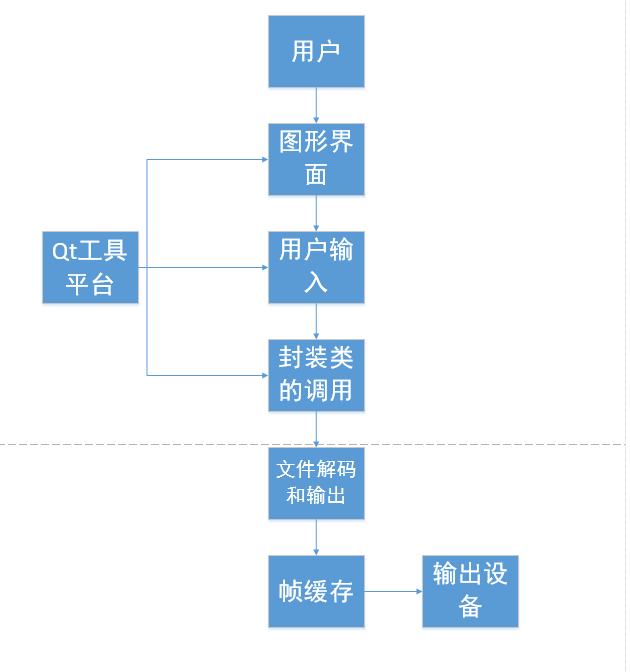


图4-1软件架构设计

**4.2播放器实现总体设计**

音视频数据先由输入输出模块,经过客户端处理后读入存储器信息。再经过读入存储器信息后的文件头,对输入输出模块执行相应的数据处理,并得到相关的信息,包括:播出时文本的编码形式、文本的尺寸大小等信息。然后确定播放器能否用于播出此文本。如可以,则使用函数连接的调用隔离器对音视频信息加以隔离。然后音频数据流和视频信号流将依次经过声音排队和录像队列,完成排队等待。然后再经过相应的声音、视频解码器,完成解码运算。最后再对经过编码后的录像信息与录像信息执行合并处理,完成同时传输和缓冲,然后再依次由相应模块输入出口。使用时,所有音视频数据流的读写、编号、输入输出都是在Qt中采用多线程的方法来完成的,即在传输的时候一直读取数据流到缓存中完成编号,然后再利用缓存的作用来增加传输的质量[10]。而因为回放信息的形式根据各个所用的解码器而有所不同,在输入模块中读出了确定的回放形式之后。就需要形式和解码器相符,方可实现打开。但在输入模块中则是直接完成了对信息的读写,而最后也将原来的音视频信息都带到了缓存中。而真正获得这些信息的需要经过接口函数调用译码器完成的[12]。

应用可以根据界面操作选项对放映文档执行开机控制的命令,根据用户输入功能接受用户操作数据来获取放映文档的文档名、档案格式等基本内容。同时检测文档名或者档案格式等内容是不是正确,查看放映文档中有无空挡文档等。随后,函数接口才刚刚开始通过调整启动解码器完成解码动作。再利用函数接口的使用初始化函数对播放器完成初始化,为视频文件的编码与回放做好了先期准备。

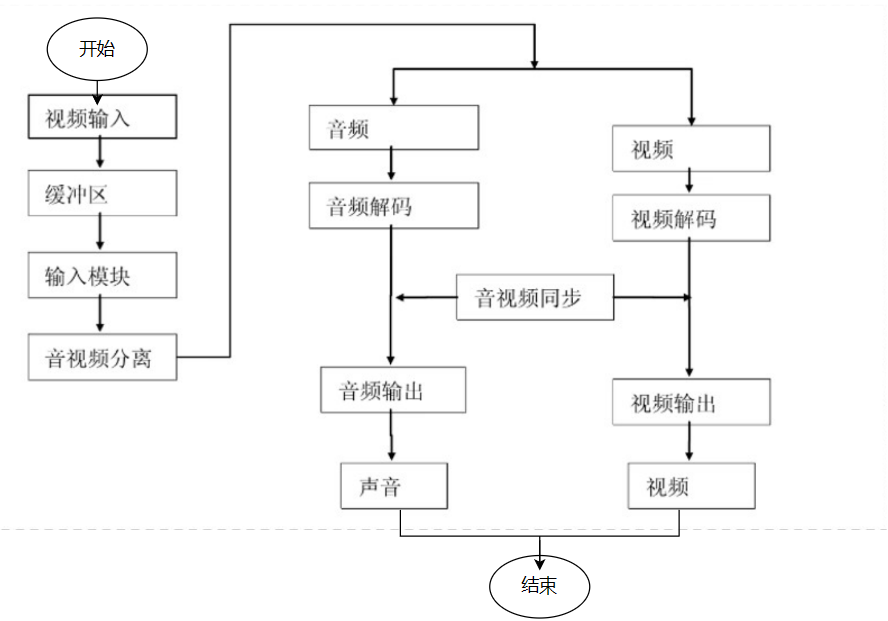


图4-2音视频播放系统流程

**4.3功能接口设计**

功能是整个播放器系统的一个关键,各个功能都有一个具体的特性,对应了一定的功能接口,里面包括了所有用户提供的功能参数。在应用层实现了底层使用时具有一个通道的功能。同时,还联系上了输入和输出模块,使数据流信息能够从输出中传递,提高了播放信息的流动性和连续性[11]。

表4-1 函数接口表

|  |  |  |  |
| --- | --- | --- | --- |
| 函数接口 | 返回值 | 参数 | 功能说明 |
| quick | void | bool flag | 实现快进快退 |
| connectDatabase | void |  | 链接数据库 |
| mouseMovevent | void | QMouseEvent\* e | 鼠标事件 |
| mousePressEvent | void | QMouseEvent\* e | 鼠标按下事件 |
| mouseReleaseEvent | void | QKeyEvent\* e | 鼠标抬起事件 |
| keyPressEvent | void |  | 按键事件 |
| On\_shrinkBtn\_clicked | void |  | 列表收缩 |
| On\_extendBtn\_clicked | void |  | 列表展开 |
| On\_volBtn\_clicked() | void |  | 音量控制 |
| On\_clearBtn\_clicked | void |  | 清楚列表 |
| On\_onLineBtn\_clicked | void |  | 在线切换列表 |

**4.4界面设计**

一个好的界面设计能够让用户快速地找到想要的功能，提高用户体验。在视频播放器的界面设计中，需要考虑到用户的习惯和需求，以及美观和易用性的平衡。本文将介绍基于Qt的视频播放器界面设计，通过使用Qt的丰富功能和灵活性，实现一个具有高度可定制性和易扩展性的视频播放器界面。

(1)界面布局

根据本程序设计的目的，程序ui界面需要有以下部分：主窗口，播放列表；三个窗口按建：最小化按钮，窗口化/最大化按钮，关闭按钮；四个播放按键：停止按键，暂停/播放切换按键，上一曲，下一曲；打开文件。

(2)主题色彩

考虑到大众的习惯，主体采用黑色。

(3)图标设计

一个清晰明了的图标是界面设计的关键，我希望采用简洁明了的图标，颜色采用白色，与背景想对比。

4.5库文件功能设计

库文件的功能设计是指为实现特定的功能或解决特定的问题而开发的库文件中所包含的一系列功能。通过良好的功能设计，可以使库文件具有更高的可重用性、可维护性、可扩展性和可靠性。

在进行库文件功能设计时，需要充分考虑用户需求、系统环境和技术实现等因素。具体来说，需要确定库文件的功能范围、接口设计、功能实现方式、错误处理机制等方面的内容。同时，还需要进行充分的测试和验证，以确保库文件的正确性和稳定性。

库文件功能设计的关键在于找到合适的抽象层次，只暴露必要的接口，将复杂的实现细节封装起来。合理设计的库文件可以提供简单易用的接口，为用户提供高效、可靠的功能操作。同时，在开发和维护过程中，良好的库文件功能设计也可以减少后续工作量和维护成本。

以下是该程序的库文件设计。

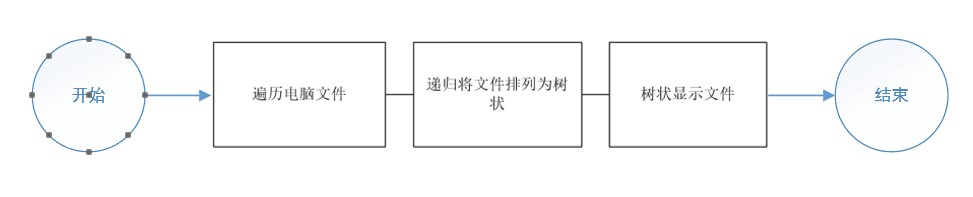
表4-2 库文件表

|  |  |
| --- | --- |
| 库文件名 | 功能 |
| BottomWidget | 按键功能 |
| FileSystemDialog | 遍历文件夹 |
| FileTreeView | 文件夹树状显示 |
| ListWidget | 播放列表 |
| MainSlider | 主窗口的播放进度条 |
| Widget | 主窗口界面 |

4.6基本功能设计

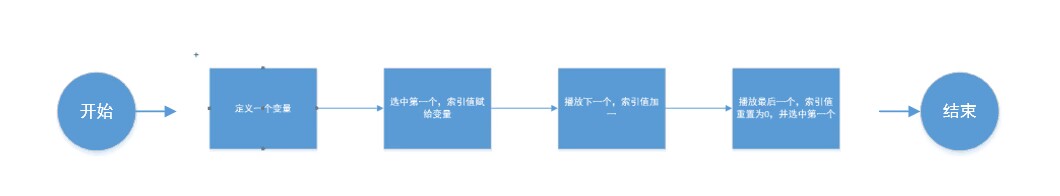
此部分主要是解决了播放器设置中关于播放器的一些基本功能的实现,一般关系到的根木操作还有停止、回放、停止、快进快退拖拉、全屏播放操作等。可以在Qt图形化设置的后台进行编程的发展,来完成这个操作。所有操作都是在Qt中使用信号与槽机制,把按钮和槽函数连接而来。重要功能代码包含:

1. 树状显示电脑文件设计



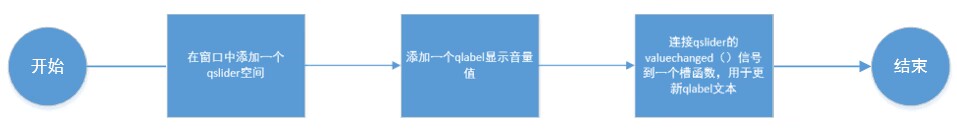
4.4.1树状显示电脑文件设计

1. 列表播放索引设计

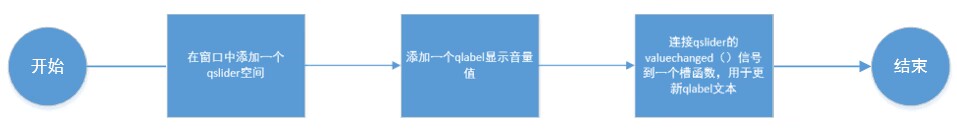


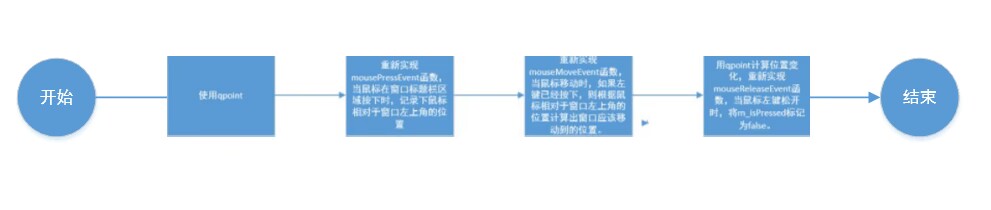
4.4.2 列表播放索引设计图

1. 音量滑块设计

4.4.3 音量滑块设计图

1. 视频进度滑块设计

4.4.4 视频进度滑块设计图

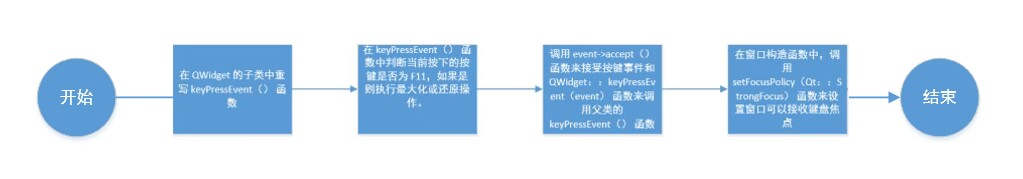
1. 暂停播放切换按键设计 4.4.5 暂停播放切换按键设计图
2. 移动窗口设计

4.4.6 移动窗口设计设计图

1. 离开窗体自动隐藏播放列表功能设计

![XY0)Q$}XQXX{G8JBOAD(](C](data:image/jpeg;base64,)4.4.7 离开窗体自动隐藏播放列表功能设计图

1. 最大化切换功能设计

4.4.8 最大化切换设计图

**5音视频播放设计测试及其效果**

该模型式此系统的重要功能模块,要实现的内容众多,例如要实现放映内容的设置、放映文档项目的转换、显示内容、终止文档、中断放映等。而在此系统中,通过QlistWidget空间也可以实现放映内容列表。通过QStackedWidget来完成观看视窗的设置,其中将音视频内容输入到视窗中的方法就是setWid类(ui.playscreen->WinId())。除了通过Qslider类来完成观看进度条的设置,在播放视频中还可拖拽进度条来完成观看过程的进退调整与选择等操作。显示过程如图5-1所显示。

经过检查可以发现播放器工作正常。启动、显示、暂停、停止、还有一个显示文件列表的选项基本功能全部实现。此外,对于显示画面的拉升、压缩和全屏播放的操作效果均良好。在显示视频文件时可实现音视频共享的功能,在显示画面动作平稳,可顺利实现快进快退和拖拉功能等。经过试验,该系统基本已实现了设计目标的要求。



图5-1最终效果



widget::widget(Qwidget \*parent) :

(Owidget(parent)，PADDING(2),clickCount(1) , isLeftPressDowmn(false)， field 'clickCount' will be initialized after field "isLeftPress0.dir(NONE)，ui(new Ui:.widget) , windowniwidth(QApplication:.desktop()->available6beometry() .width())\*29/48)，

A availablebeonetry' i.playingIndex(-1),

windowIniHeight((QApplication::desktop()->availableGeometry () .height())\*20/27)，

A 'availab.

screenMidth(QApplication::desktop()->availablebeometry ().width())， A 'availableGeometry'is deprecat.screenHeight(QApplication::desktop()->availableGeometry().height())

A 'availableGeometry' is deprec...

ui->setupUi (this);

this->setAutoFillBackground (true) ;

this->setstylesheet( "Qwidget{background-color:rgb(28,28,28);}");setwindowOpacity(1);//设置透明度

this->setMinimumHeight(500) ;this->setMinimumwidth(720);

this->setWindowFlags(Qt::FramelessWindowHint|Qt::windowSystemNenuHint);// 设置成无边框对话框this->setMouseTracking(true) ;

this->setFocusPolicy(ot: :clickFocus) ;this->setNinimumHcight(508) ;

this->setNinimumwidth(728);

this->setiindowFlags(Qt : ; FramelessWindowtint| Qt : :WindowSystenmHenuHint);//设置成无边框对话框this->setHouseTracking(true);

othis->setFocusPolicy(ot : :clickFocus);

图5-2 主界面及代码

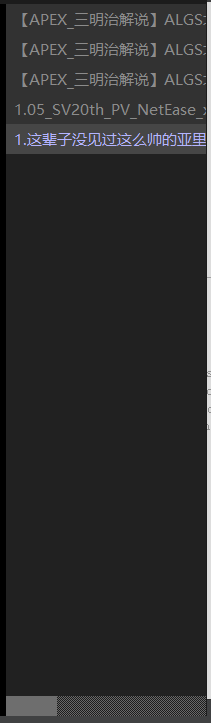
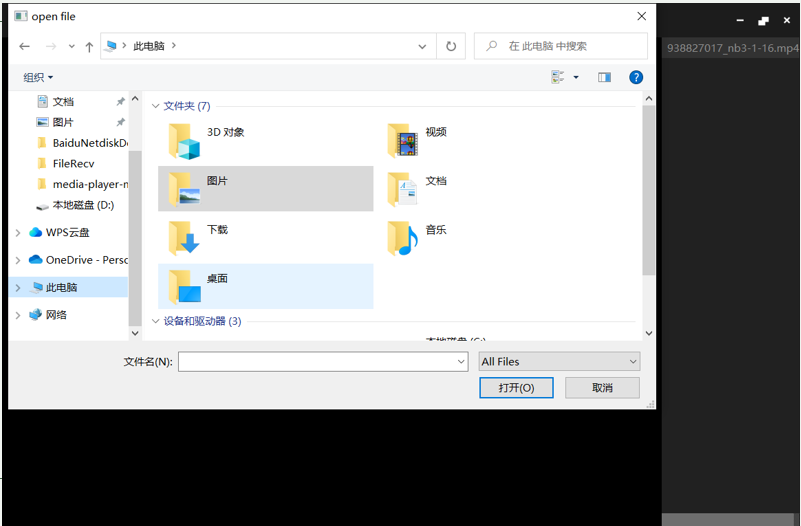


图5-3 播放列表



menuButton->setGeometry ( 10,10,30,30);

menuButton->setStyleSheet("QPushButton{ image:url(: /play .png); border-radius:15px; order: none; background-color:rgb(28,28,28);]"

"QPushButton:checked{ background-color:rgb(0,0,1);}"

"OPushButton: :menu-indicator{image:None: "

void widget: :addFile(const QString fileName)

if( !fileName.isEmpty())

Qstring name = getNameFromPath(fileName);//从路径中获取文件名字/ltitleLabel->setText(name) ;

int i;

for(i=0 ;i<vector.size() ;i++){

if(vector.at(i).first.second == fileName)//该文件已经存在{

playingIndex=i;

mediaPlayer0 ->setMedia(QUrl: :fromLocalFile(fileName)) ;mediaPlayero->play ();

listwidget->setCurrentRow(i);listwidget->setIndex (i);

titleLabel->setText(removeIndexFromAppend(vector.at(i).second)) ;break;

}

}

图5-4 打开文件

void widget: :addFile(const QString fileName)

if( !fileName.isEmpty())

Qstring name = getNameFromPath(fileName);//从路径中获取文件名字/ltitleLabel->setText(name) ;

int i;

for(i=0 ;i<vector.size() ;i++){

if(vector.at(i).first.second == fileName)//该文件已经存在{

playingIndex=i;

mediaPlayer0 ->setMedia(QUrl: :fromLocalFile(fileName)) ;mediaPlayero->play ();

listwidget->setCurrentRow(i);listwidget->setIndex (i);

titleLabel->setText(removeIndexFromAppend(vector.at(i).second)) ;break;

}

}

/从路径中得到文件名字

string widget: :getNameFromPath(const Qstring &fileName)

Qstring name;

QChar c;

QString s = fileName.toUtf8( ).data();foreach(c，s)

{

if(c =='/'){

name.clear();continue;

}

name. append(c) ;

}

l/name = name.toUtf8( ).data() ;

return name;



//下一曲

void widget: :slotNextVideo()

if( !vector.isEmpty())

if(++playingIndex == vector.size())

playingIndex=0;

listwidget->setIndex(playingIndex) ;

mediaPlayer0 ->setMedia(QUrl::fromLocalFile(vector.at(playingIndex) .first.second));mediaPlayer0->play();

listwidget->setCurrentRow(playingIndex);

}

//上一曲

void widget: :slotLastVideo(){

if( !vector.isEmpty())

if(--playingIndex <= -1)

playingIndex=vector.size()-1;listwidget->setIndex(playingIndex ) ;

mediaPlayer0 ->setMedia(QUrl: : fromLocalFile(vector.at(playingIndex).first.second));mediaPlayero->play (;

listwidget->setCurrentRow(playingIndex) ;

//停止视频

void widget: :slotStopVideo(){

if( ( !vector.isEmpty() )&&(mediaPlayerO->state() !=QMediaPlayer::StoppedState))

mediaPlayer0->stop();

mainLabel->clear;mainLabel->show();

图5-5 停止，上一曲，暂停，下一曲



/滑动条

slider = new Mainslider(Qt : :Horizontal，bottomwidget);slider->setRange(0,0);

'slider->setGeometry( 10，7，normalRect.width()-225，13);

connect(slider，SIGNAL(sliderMoved(int))，this，sLOT(slotSetValue(int)));

图5-6 进度滑块



//音量技钮

volummeButton = new QPushButton(bottoalwidget;

volumeButton->setGcometry(bottomMidget->width( )-88,35,30,30);

volumeButton->setStyleSheet("QPushButton{inage1url(1/volume .png);border-radiust15px; backround-color: rgb(0,8,6)]"

“QPushButton::hover[ inage:url(: /volumne .png) ; order-color:lpx black; background-color:rgb(54,54,54);)")

volumewidget = new Qwidget(this);

volumeButton->installEventFilter(thi=);

volumewidget->setwindowFlags(Qt: : FranelesswindowHint /Qt: :window5ystenMenuHint);volumewidget->setstylesheet ( "background-color:rgb(25，38，58)");

wolumcwidget->setFixedwidth(30y;

wolumawidget->installEventFilter(this);

wolumeslider = new volume5lider(Qt::vertical,this);

volumeslider->setGeometry(volumewidget->width()-80,35,30,30);wolumeSlider->setRange(0,i00) ;

wolumeSlider->setCursor(Qcursor(Qt:t PaintingHandCursar)) ;

connect (volumeSlider，SIGNAL(valueChanged(int))， mediaPlayer0，SLOT(setVolume( int)));//响应声音滑动条的改变connect(volumeSlider，SIGNAL(sliderWoved(int))， mediaPlayer6，SLOT(setVolume( int)));//响应声音滑动条的移动

connect(volumeSlider，SIGNAL(signalPressVolumePosition(qint64))，this，sLOT(slotPressVolumePosition(qint64)));

图5-7 音量滑块



/设置右上角的三个按锰

oid lwidget: :setThreeButton()

设置三个孩钮

eloseButton-new QPushButton(this);nindmunButton

rewm QPushEutton(this) ;maximaanButton - new QPushButton(this;closeButton->setFixedS百zel3e, 28);

was imunButton->setFixedSize(3e,20);ninimumButton-3cetFixedsize(38, 20);

8int a =thio->rectO - topRTghtl).×(;int a - nornalRect.widtho;

closeButton->setGeonetey(a-34，15，20，15);//行关闭按妞故在最右上乌maximumButton->setGeonetry(a-68,15，20，15);ninimaunButton->setGeometry(a-102, i5,，20，15);l/按钮之间闻香6p工QString name - QD1r::currentPath() ;

eloseButton->setStyleSheet(""QPushButton{inoge:url(:/close.png);

border-ratius:&px; background-color'rgb(28,286,28);}"

“QPushButton:hover{[background-color:rgb(54 ,54, 54) ; image-url(: /close.png); "");

minimunbutton->setstyleSheet(QPushButton( imnoge:url(:/mininize.png);

border-radfus:dpx; bsckground-color-'rgb(26,28,28);]"’

" “rpPushbutton:hover{inage url(ainfnfzetivered.pag);

backround-cor:rg)(54 54,a4);tmsgeurl(: /mtntnfte.an); 7;

nowimnbuttonrstStyleShet(""Purhluttonfinoge;url( /naxinite pn) bordorrodivns-tp; borckgreond color'rgb(21,12,1B);)" n pn

QPushBbutton:hover{fmaget:url(naxinfizetiovered.ng) ;

bacxground-colorirgl(54,54,54); sge:urt(:/naximnize.,ang); );

eLo卷eEutton->setToolTip(""关闸");

ninimuwButton->setToalTip(”最小化"";

naximumButton-3cetToolTp(“最大化/迁京""3;

/eonnect(eloseButton，SIGNAL(e1ieked() )，this，SLOT(elose() ) );

connect(closeButton，SIGNAL(clicked()) , this,SLOT(slotClosewidget0));

connect(maximunEutton，SIGNAL(clicked()，this，SLOT(slotSetButton()>);eonneet(minimusButton，STIENAL(elicked(5)，this，SLOT(showNininized0O) );

,

图5-8 最小化，最大化和关闭

**总结**

本文基于Qt框架，实现了一款简单的视频播放器。该播放器支持多种视频格式的播放，包括常见的MP4、AVI、FLV等格式。通过使用Qt提供的多媒体模块，实现了视频的解码、播放、暂停、停止等功能，并且支持实时的视频截图和全屏播放。

在实现过程中，本文使用了Qt的信号与槽机制，实现了播放器控件与视频播放器之间的交互。同时，本文也对Qt的多线程机制进行了研究，使用了多线程技术来提高视频的解码、播放效率，保证了视频的流畅播放。

整个播放器的界面设计简洁明了，主要分为播放器窗口、播放列表窗口和截图窗口三个部分。播放器窗口主要用于显示视频画面，提供了播放、暂停、停止、全屏等常用的操作按钮；播放列表窗口用于管理多个视频，支持添加、删除、播放等操作；截图窗口则用于实时显示当前视频帧的截图，用户可以通过截图窗口保存自己喜欢的视频截图。

总之，本文设计的视频播放器基于Qt框架，实现了多种视频格式的播放，功能完善，界面简洁明了，具有一定的实用价值。