重庆师范大学

QT课程设计

`

**项目名称： 视频剪辑器开发**

**专业年级： 软件工程2022级**

**小组成员： 张渝 谢雪莲 郑豪**

**指导教师： 龚伟老师**

**2024** **年** **07** **月 7** **日**

**目录**

[一、项目介绍 2](#_Toc2147061964)

[二、研发任务分配 2](#_Toc1621227464)

[三、问题分析处理 2](#_Toc1490686088)

[1、 出现的问题以及解决方法 2](#_Toc876769592)

[2、遗留问题 3](#_Toc98259209)

[四、项目分析设计 3](#_Toc2103680653)

[1、软件介绍 3](#_Toc1520707833)

[2、 设计 4](#_Toc87074338)

[3、UI设计 9](#_Toc1241854951)

视频剪辑器开发文档

## 一、项目介绍

该视频剪辑器是依据VidCutter提供的功能、界面模仿开发。软件提供了简单的视频播放暂停、选择视频源文件、剪切视频片段和合并并保存所有剪切片段。额外添加了提取视频内字幕流（如果视频本身没有ffmpeg支持的字幕流，则不能提取）和为视频添加字幕流功能。在添加字幕时，选择了多进程编程，防止系统出现假死的情况。在涉及唤出对话框，并与后续功能实现有明显的同步需求时，利用信号与槽的机制实现了从互斥到同步的实现。

## 二、研发任务分配

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **任务时间** | | | | | |
| 开始时间： | | 2024.6.17 | | 结束时间： | 2024.7.8 |
| **任务说明及分配** | | | | | |
| **编号** | **执行者** | | **任务详细说明** | | |
| 1 | 张渝（组长） | | 开发文档总编辑、分配任务、调整进度与组员协调、组员出现问题时调试bug；字幕流功能的提供、提供涉及时间处理的接口、大部分JS代码、所有按钮在前端和部分后端的交互以及接口、进度条（Timer）全部设计、前端总体框架设计、视频播放相关设计 | | |
| 2 | 谢雪莲 | | 初步实现了视频合并功能，完成了对视频剪辑路径的保存，实现了素材列表对剪辑路径的获取、完成了素材列表add和remove的实现、重构ui的结构、小部分JS代码、前端组件的编写和修改、注册C++对象到QML中、资源文件的引入 | | |
| 3 | 郑豪 | | 实现了视频剪辑功能，以及完成了视频剪辑，视频合并的接口，实现了前端素材列表的ui设计，以及完善了视频剪辑功能和视频合并功能，添加了临时存放裁剪的视频和存放路径的文本文档的目录 | | |

## 三、问题分析处理

### 出现的问题以及解决方法

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 问题描述 | 解决方案 | 解决人 |
| 1 | 最开始的时候，状态栏是用Rectangle实现的，会与其他组件发生重叠的情况 | 将状态栏在Window中用footer重新实现并调整所有与之有交互按钮的onHover | 张渝 |
| 2 | 调用对跨框后，无法为后续有同步需求的代码提供同步。还涉及连续多次对话框调用，均有同步要求 | 使用信号与槽实现从互斥到同步的转换。涉及连续的同步，则以是否按下确认键为信号，进行编程。 | 张渝 |
| 3 | 打开这个软件后裁剪一个视频后关闭软件再打开这个软件会出现剪辑会出现崩溃 | 因为视频存储的位置相同，且命名相同所以会出现崩溃，所以每关闭软件之后就会删除截取的视频 | 郑豪 |
| 4 | 起初裁剪的视频只能放在播放视频所在的目录下且关闭软件后裁剪的视频不能删去 | 打开一个软件后生成一个目录，将裁剪的视频以及存储裁剪视频的路径的文本文档保存在里面，关闭软件后删去这个目录 | 郑豪 |
| 5 | QML的url到C++的String类型转换 | 最开始的思路是：在videoEdit中设置string url接受这些url,然后传给videocut的in\_filename，在从out\_filename将他保存到一个一个文档中，用于videoMerge的fileList调用。但是后来发现可以直接使用QString类型，自动实现Qwidget与QML的类型转换 | 谢雪莲 |
| 6 | 资源文件的载入 | 绝对路径可以用，相对路径不能用。最后发现在cmakelist.txt中少了一个宏的定义set(CMAKE\_AUTORCC ON) | 谢雪莲 |
| 7 | remove要实现两个地方的删除：文件filepath.txt清空和ListView清空。只能实现ListView的清空，但是无法清除完filepath.txt的内容，始终会有file ‘’存在 | 在C++写了remove(),直接在这里面找到文件路径并清空文件的内容 | 谢雪莲 |
| 8 | 不能暂停视频问题（pause） | 改变Timer的value值，使得在当前位置positionUpdateTimer.running将他的状态改变成false。就能够暂停了，之后再点击一次，就能将他的转台改变回true | 谢雪莲 |
| 9 | 在没有视频的前提下点击字幕功能，也会进行响应 | 在Action中添加状态位，然后在相关交互处理中，判断enable是否为true,若判断成功再进行下一步；否则停止 | 张渝 |

`

### 2、遗留问题

|  |  |  |
| --- | --- | --- |
| 编号 | 问题描述 | 大致解决方案 |
| 1 | 添加字幕流后，能在系统自带播放器如Haruna中显示字幕文件，但是不能在软件播放器中显示。即软件无法显示字幕 | 用c++开发独立的视频字幕显示组件，并重构添加字幕的功能实现代码：不是通过为视频添加字幕流（依赖视频播放器的功能提供），而是通过独立的c++组件完成对字幕流文件的解析并显示。 |
| 2 | 点击remove之后，能够删除素材列表项的内容，但是再裁剪一次。filepath.txt里面有路径，但是ListView却无法显示当前列表项。 | 在QML的.js文件里写了一个add（）函数，可以让列表项重新展示。但是它不仅会展示当前裁剪的视频项，还会展示以前被remove的视图项，但是展示出以前的视图项里面是没有路径的。只有在add之后才会出现路径。 |
| 3 | 原定在进度条上有一个指针跟随进度。编写完代码后虽然能实现移动动画，但不能实现跟随，走得太快了 | 暂无思路 |
| 4 | 进度条中的播放跳转性能不足：使用Timer就不能实现单次点击跳转；不使用Timer就会导致视频播放异常卡顿。 | 用FFmpeg或QML底层实现一个视频的播放器，代替Qt本身自带的MultiPlayer. |

## 四、项目分析设计

### 1、软件介绍

本项目是一款基于QML与FFmpeg开发的视频剪辑器。剪辑器剪辑下的视频片段和最终合并的视频会保存在一个目录中，可供用户自行使用。

在进行字幕功能时，请先保证当前有视频正在播放，无论是提取还是添加字幕，都是基于当前正在播放的视频进行的操作。在完成字幕添加功能时，会生成一个将当前视频与字幕文件合并后的新视频，原本设计中在完成字幕合并后，软件会自动更换视频源到新的视频，也做好了从c++端到QML的信号处理（MainContent中末尾的Component），但是并没有成功。

### 设计

##### **2.1 概要设计**

功能概要设计如下：

1. 选择视频源：通过Content中MyRadioButton的一个实例 \_openfile 的onTapped调用Controller中的openfileTriggered函数，打开Dialog中\_openfile对话框选择。然后在MainContent的Dialogs实例中调用onAccepted实现同步调用setfilepath设置视频源
2. 视频播放： 通过Content中MyRadioButton的一个实例 \_play,判断MultyPlayer的播放状态并进行播放暂停的切换
3. 剪切开始： 通过Content中MyRadioButton的一个实例 \_startcut,调用Controller中的startcutTriggered()获取当前相对于视频起点的时间戳。
4. 剪切结束：通过Content中MyRadioButton的一个实例 \_endcut,调用Controller中的endcutTriggered()获取当前相对于视频起点的时间戳。然后调用c++注册类实例videoeidt中videocut函数，通过FFmpeg进行剪切。
5. 合并并保存：通过Content中MyRadioButton的一个实例 \_save,调用Controller中的saveTri2ggered()获取生成文件名即该文件名的绝对路径，然后调用video中的Merge函数实现合并并保存的功能。
6. 提取字幕： 在有视频播放的前提下，点击工具栏中Subtitle中 get subtitle file,然后通过Controller调用video中getSubtitle函数对正在播放视频的字幕流文件进行提取
7. 添加字幕： 在有视频播放的前提下，点击工具栏中Subtitle中 addsubtitle file,然后通过Controller调用video中addtSubtitle函数对正在播放视频添加（如果有字幕流则会覆盖）指定的字幕流文件。该字幕文件可以精确到毫秒进行自定义。
8. 删除素材列表：再有剪切视频片段的前提下，点击remove会通过Controller调用c++的remove（）实现文件内容的清空，并更新素材列表项的内容，将其所展示的内容也清空。
9. 添加素材列表： 在删除素材列表之后，再裁剪视频片段无法显示，点击add回通过Controller的addTriggered（）实现素材列表项的显示。

##### **2.1 详细设计**

1. 选择视频源：通过Content中MyRadioButton的一个实例 \_openfile 的onTapped调用Controller中的openfileTriggered函数，打开Dialog中\_openfile对话框选择视频路径。

然后在MainContent的Dialogs实例中调用onAccepted实现同步调用setfilepath，在确认点击确定后，设置视频源，将视频区初始背景改为不可见并开始播放。

1. 视频播放： 通过Content中MyRadioButton的一个实例 \_play,判断MultyPlayer的播放状态playbackRate并进行播放暂停的切换
2. 剪切开始： 通过Content中MyRadioButton的一个实例 \_startcut,调用Controller中的startcutTriggered()获取当前相对于视频起点的时间戳。

首先设置按钮\_startcut和\_endcut的enable状态值（如果状态值为false,那么此步无法进行），然后通过MainContent中MultiPlayer的position值获取当前时间戳，并返回

1. 剪切结束：通过Content中MyRadioButton的一个实例 \_endcut,调用Controller中的endcutTriggered()获取当前相对于视频起点的时间戳。然后调用c++注册类实例videoeidt中videocut函数，通过FFmpeg进行剪切。

首先设置按钮\_startcut和\_endcut的enable状态值（如果状态值为false,那么此步无法进行），然后通过MainContent中MultiPlayer的position值获取当前时间戳，并返回

VideoEdit中的videocut函数：启动ffmpeg调用相关视频裁剪的命令，然后将裁剪的视频存放到代码相关路径的子目录temp目录下

1. 合并并保存：通过Content中MyRadioButton的一个实例 \_save,调用Controller中的saveTri2ggered()获取生成文件名即该文件名的绝对路径，然后调用video中的Merge函数实现合并并保存的功能。

VideoEdit中的Merge函数：首先创建一个临时文件保存filepath.txt的绝对路径，然后使用ffmpeg执行合并视频文件的命令：ffmpeg -f concat -safe 0 -i filepath.txt -c copy videoMerge.mp4合并filepath.txt里面的路径

其中 **-f concat**: 指定输入文件的格式为concatenate（连接）。要使用concatenate协议来处理输入。

**-safe 0：**关闭安全模式。如果文件路径不在当前目录下，FFmpeg默认会拒绝处理，除非设置了-safe 0来关闭此安全检查。

-i:将filepath.txt里面的路径连接起来处理

-c copy:这个参数指定使用“copy”编解码器，意味着视频流和音频流将直接复制到输出文件，而不进行重新编码。这样可以保留原始的编解码格式和质量，同时加快处理速度和避免质量损失。

6提取字幕： 在有视频播放的前提下，点击工具栏中Subtitle中 get subtitle file,然后通过+Controller调用video中getSubtitle函数对正在播放视频的字幕流文件进行提取。

JS的getSubtitle函数中，获取需要的路径参数，然后调用VideoEdit中的getSubtitle函数

VideoEdit的getSubtitle函数中，指定ffmpeg执行文件的位置，输入命令

ffmpeg -i inputFile.mp4 -map 0:s:0 output\_subtitle.srt。

其中 -i指定输入文件的文件路径，-map指定字幕流，最后指定提取出的字幕文件的保存路径。

7添加字幕： 在有视频播放的前提下，点击工具栏中Subtitle中 addsubtitle file,然后通过Controller调用video中addtSubtitle函数对正在播放视频添加（如果有字幕流则会覆盖）指定的字幕流文件。该字幕文件可以精确到毫秒进行自定义。

JS的addSubtitle函数中，因为涉及两个异步对话框，所以要通过信号与槽进行同步设置。先在获取当前视频路径、添加的字幕文件；在确认字幕文件的路径后，通过信号与槽打开保存合并后视频路径的对话框；在确认视频路径后，调用VideoEdit中addSubtitleAsync函数。

VideoEdit的addSubtitleAsync函数中创建了新的子进程调用addSubtitle函数，并设置了相关的子进程销毁函数。在子进程结束任务后发送信号sunfinished,在MainContent中进行同步信号处理，使当前视频播放源改为被合并视频的路径（并未成功）。

VideoEdit的addSubtitle函数，指定ffmpeg执行文件的位置，输入命令

ffmpeg -i inputMediaFile.mp4 -i subtitleFile.srt -c:v libx264 -c:a aac -c:s mov\_text -map 0:v -map 0:a -map 1 -metadata:s:s:0 language=eng -disposition:s:0 default ouput2.mp4

以下为指令解释

-c:v libx264:

-c:v 指定了视频编码器，这里使用了 libx264，它是H.264编码器的库。

libx264 是FFmpeg中用于H.264编码的库，通常与 -preset medium 或 -preset slow 一起使用来优化编码速度和压缩率。

-c:a aac:

-c:a 指定了音频编码器，这里使用了 aac，它是一种高效的音频压缩格式。

-c:s mov\_text:

-c:s 指定了字幕编码器，这里使用了 mov\_text，它是一种将字幕嵌入视频文件中的格式，通常用于QuickTime文件。

-map 0:v:

-map 参数用于指定输入文件中的流映射。这里，0:v 表示映射输入文件中的视频流（通常是 Stream #0:0）到输出文件中。

-map 0:a:

同样，-map 0:a 表示映射输入文件中的音频流（通常是 Stream #0:1）到输出文件中。

-map 1:

-map 1 表示映射输入文件中的第二个流（通常是 Stream #0:2）到输出文件中。这通常是字幕流。

-metadata:s:s:0 language=eng:

-metadata 参数用于添加元数据到输出文件。这里，s:s:0 指定了字幕流，language=eng 指定了字幕的语言为英语。

-disposition:s:0 default:

-disposition 参数用于设置字幕的显示方式。这里，s:0 指定了字幕流，default 表示字幕应该始终显示。

ouput2.mp4:

这是输出文件的名称，指定为 ouput2.mp4。

1. 删除素材列表项：再有剪切视频片段的前提下，点击remove会通过Controller调用c++的remove（）实现文件内容的清空，并更新素材列表项的内容，将其所展示的内容也清空。
2. 显示素材列表：在使用了remove之后，点击add会通过Controller调用addTriggered（）显示素材列表项

#### 2.2、QML介绍

##### 2.1.1、QML功能说明

1、Window:c++引擎加载的QML，提供了工具栏、Content和状态栏Footer.。

2、Action：为工具栏中对字幕功能提供交互按钮。

3、Content： 界面设计。包括视频播放（MainContent）、剪切片段提示（MySquareButton）、进度条（Slider）、交互按钮（MyRadioButton）。

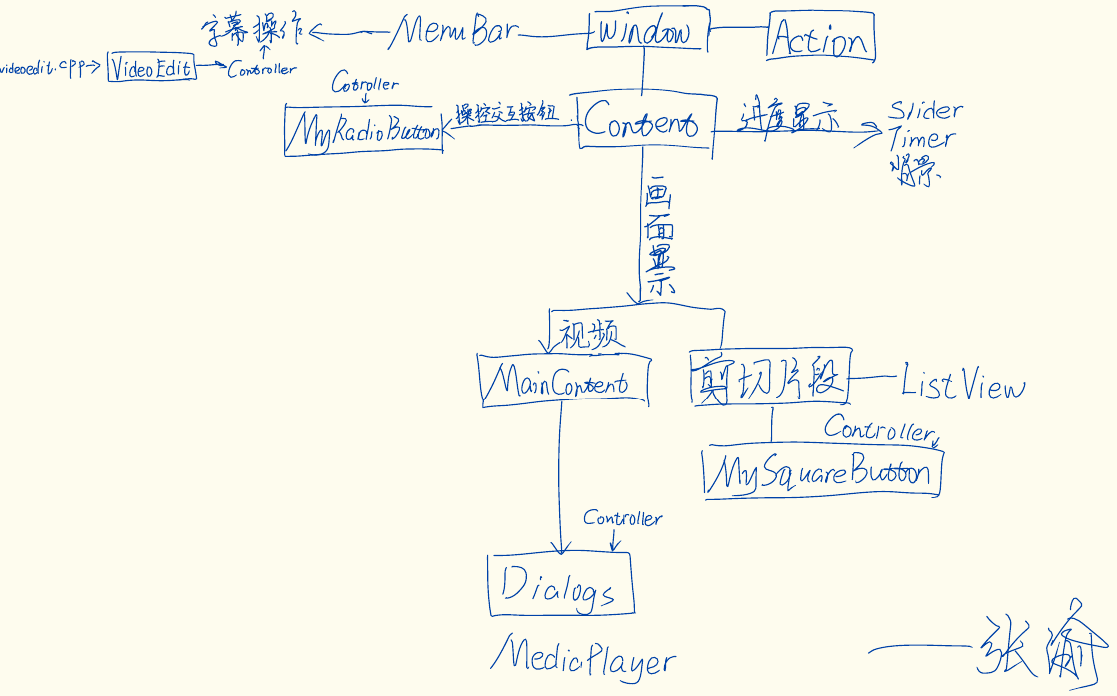
4、MainContent：实例化Dialogs、VideoEdit,为没指定视频路径前添加背景图片，添加MultiPlayer并设置相应的槽函数与事件处理器

5、MyRadioButton：为视频相关功能提供交互按钮的基类

6、MySquareButton： 为剪切视频片段提供交互按钮的基类

7、Dialogs： 合并所有对话框。

#### 2.2、QMl中大致的调用关系



### 3、UI设计

