**个人报告**

汪克杰 3130101964

1. **主要工作**
2. 音乐播放器的各项功能的设计与底层的实现。主要包括player.h和player.cpp中的代码
3. 完成ViewModel层中的各个Command的实现的函数的body部分
4. 完成对于player类的Debug并且协助完成各个模块的拼接工作
5. **实现平台**
6. **操作系统**：Linux
7. **编写平台：**Qt5.4及以上版本
8. **实现原理**
9. **Qt类库的选择以及其简单介绍**

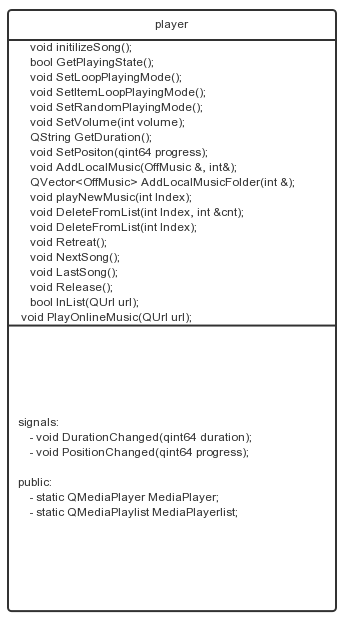
此音乐播放器的实现采用的平台为Qt5.4及其以上的版本，实现主要基于Qt的多媒体库中的QMeidaPlaylist和QMediaPlayer库。以下为Qt Document中关于这两个类的介绍：

**QMediaPlayer** The QMediaPlayer class allows the playing of a media source. The [QMediaPlayer](http://doc.qt.io/qt-5/qmediaplayer.html) class is a high level media playback class. It can be used to playback such content as songs, movies and internet radio. The content to playback is specified as a [QMediaContent](http://doc.qt.io/qt-5/qmediacontent.html) object, which can be thought of as a main or canonical URL with additional information attached. When provided with a [QMediaContent](http://doc.qt.io/qt-5/qmediacontent.html) playback may be able to commence.

**QMediaPlaylist** The QMediaPlaylist class provides a list of media content to play. [QMediaPlaylist](http://doc.qt.io/qt-5/qmediaplaylist.html) is intended to be used with other media objects, like [QMediaPlayer](http://doc.qt.io/qt-5/qmediaplayer.html). QMediaPlaylist allows to access the service intrinsic playlist functionality if available, otherwise it provides the local memory playlist implementation.

1. **底层的实现原理**

player类的主要是为音乐播放器的各个功能在底层的具体实现，包括音乐的播放和暂停，播放器的音量的设置以及播放列表中音乐的增加删除，播放模式的切换和上（下）首,获得当前的歌曲的播放时间和播放状态，以及对于歌曲的播放进度的获取以及设置。



**图一 player类UML图**

1. **实现过程**
2. **播放器的初始化以及列表存储**

由于此音乐播放器实现了对于用户上次的播放列表的存储功能，因此此提供了对于播放列表的初始化以及最后关闭播放器的写入功能，具体的实现为何淂劲同学实现，此仅仅是对函数的调用。

* Initialize：初始化播放器。主要指播放列表的初始化以及播放器声音及播放模式初始化
* Release：更新播放列表。主要是讲用户更新过的列表同步到本地以便下次使用

|  |
| --- |
| void Player::Initialize(QVector<OffMusic> &qvec){  MediaPlayerlist.setPlaybackMode(QMediaPlaylist::Loop);  if(!&MediaPlayerlist)  qDebug() << "NULL";  //set again  MediaPlayer.setPlaylist(&MediaPlayerlist);  MediaPlayer.setVolume(50);  //initilizeSong();  InitMediaList(&MediaPlayerlist, qvec);  }  void Player::Release(){  SyncMediaList(&MediaPlayerlist);  } |

1. **播放列表的增加和删除歌曲**

用户可以选择从添加一首歌曲或者从文件夹中添加，然后删除播放列表中的一首歌

* AddLocalMusic: 添加单首歌曲。从本地文件中添加一首歌曲到播放列表
* AddLocalMusicFolder: 从文件夹添加歌曲。将本地文件夹中的所有歌曲添加到播放列表
* DeleteFromList：删除歌曲。从播放列表中删除一首歌曲

|  |
| --- |
| void Player::AddLocalMusic(OffMusic &offmusic, int &index)  {  //add to list  QUrl fileurl = QFileDialog::getOpenFileUrl(0, QObject::tr("Open Music File"), QObject::tr("."), QObject::tr("mp3 music files(\*.mp3)"));  if(InList(fileurl) || fileurl.isEmpty())  {  throw AlreadyInListException();  qDebug() << "media count " << MediaPlayerlist.mediaCount();  }  if(!MediaPlayerlist.addMedia(fileurl))  throw AddToListException();  qDebug() << "media count " << MediaPlayerlist.mediaCount();  OffMusic omus(fileurl);  offmusic = omus;  index = MediaPlayerlist.mediaCount();  qDebug() << "index" << index;  }  QVector<OffMusic> Player::AddLocalMusicFolder(int &MusicCount)  {  MusicCount = MediaPlayerlist.mediaCount();  qDebug() << "media count " << MediaPlayerlist.mediaCount();  QUrl url = QFileDialog::getExistingDirectoryUrl(0, QObject::tr("Open Music File Folder"));  QDir dir(url.toLocalFile());  QStringList filter;  filter << "\*.mp3";  QFileInfoList list = dir.entryInfoList(filter);  QVector<OffMusic> qvec;  foreach(QFileInfo info, list)  {  QUrl fileurl = QUrl::fromLocalFile(info.absoluteFilePath());  if(!InList(fileurl))  {  if(!MediaPlayerlist.addMedia(fileurl))  throw (AddToListException());  qDebug() << fileurl;  OffMusic omus(fileurl);  qvec.append(omus);  }  }  int cnt = MediaPlayerlist.mediaCount();  for(int index = 0; index < cnt; index++)  {  qDebug() << MediaPlayerlist.media(index).canonicalUrl();  }  qDebug() << "media count " << MediaPlayerlist.mediaCount();  return qvec;  }  void Player::DeleteFromList(int Index, int &cnt)  {  qDebug() << MediaPlayer.state();  int count\_begin = MediaPlayerlist.mediaCount();  for(int i = 0; i < count\_begin ; i++)  qDebug() << MediaPlayerlist.media(i).canonicalUrl();  //MediaPlayerlist.setCurrentIndex(-1);  qDebug() << "delete index " << Index;  qDebug() << "current index" << MediaPlayerlist.currentIndex();  qDebug() << "current size" << MediaPlayerlist.mediaCount();  if(MediaPlayer.state() == QMediaPlayer::PlayingState)  {  MediaPlayer.deleteLater();  MediaPlayer.pause();  if(!MediaPlayerlist.removeMedia(Index))  {  qDebug() << "remove fail";  throw (DeleteFromListException());  }  }  else  {  if(!MediaPlayerlist.removeMedia(Index))  throw (DeleteFromListException());  }  }  } |

1. **播放器的控制功能实现**

此部分为播放器的控制功能的实现，主要包括以下的一些功能：

* playNewMusic: 播放一首新的歌曲。传入播放列表中的位置index，从头播放该首歌曲
* playOnlineMusic: 播放网络歌曲。传入网络歌曲的url，播放该首歌曲
* Retreat： 播放（暂停）键。当前播放歌曲则暂停，暂停则继续播放，无歌曲播放第一首
* NextSong/LastSong：上一首/下一首。根据播放模式播放下一首歌曲
* SetPosition: 播放进度的调整。根据用户调整的播放进度调到歌曲的该进度
* SetVolume：播放器声音的设置。根据用户设置的声音调整播放器播放的声音
* SetPlayingMode：设置播放模式。更换播放器的播放模式（循环，随机，单曲循环）

|  |
| --- |
| void Player::playNewMusic(int Index){  qDebug() << "index in play new" << Index << endl;  if(Index >= MediaPlayerlist.mediaCount())  {  return;  }  qDebug() << "before set" ;  // MediaPlayer.setPlaylist(&MediaPlayerlist);  MediaPlayerlist.setCurrentIndex(Index);  qDebug() << "index in play new after set" << 、MediaPlayerlist.currentIndex();  qDebug() << MediaPlayer.currentMedia().canonicalUrl();  QUrl url = MediaPlayer.currentMedia().canonicalUrl();  if(!QFile::exists(url.toLocalFile()))  {  try  {  DeleteFromList(Index);  }  catch(const DeleteFromListException &deletefromlistexception)  {  throw;  }  throw playNewMusicException();  }  else  MediaPlayer.play();  }  void Player::PlayOnlineMusic(QUrl url)  {  qDebug() << "set before" << url;  MediaPlayer.setMedia(url);  qDebug() << "after set" << url;  MediaPlayer.play();  qDebug() << "play online music" << url;  MediaPlayer.setPlaylist(&MediaPlayerlist);  }  void Player::Retreat(){  //no music is playing and playing list is not null and playing first music  if(MediaPlayer.state()==QMediaPlayer::StoppedState && !MediaPlayerlist.isEmpty())  playNewMusic(0);  //music is playing and  else if(MediaPlayer.state()==QMediaPlayer::PausedState)  MediaPlayer.play();  else if(MediaPlayer.state()==QMediaPlayer::PlayingState)  MediaPlayer.pause();  qDebug() << "Retreat";  }  void Player::NextSong(){  QUrl url = MediaPlayer.currentMedia().canonicalUrl();  if(!url.isLocalFile())  return;  int NextIndex = MediaPlayerlist.nextIndex();  playNewMusic(NextIndex);  }  //last song  void Player::LastSong(){  QUrl url = MediaPlayer.currentMedia().canonicalUrl();  if(!url.isLocalFile())  return;  int PreviouseIndex = MediaPlayerlist.previousIndex();  playNewMusic(PreviouseIndex);  }  void Player::SetPositon(qint64 progress)  {  //qint64 position = percent\*1.0/100 \* GetDuration();  MediaPlayer.setPosition(progress);  }  void Player::SetVolume(int volume){  MediaPlayer.setVolume(volume);  }  //set the playing mode loop  void Player::SetLoopPlayingMode(){  MediaPlayerlist.setPlaybackMode(QMediaPlaylist::Loop);  }  //set the playing mode sequential  void Player::SetItemLoopPlayingMode(){  MediaPlayerlist.setPlaybackMode(QMediaPlaylist::CurrentItemInLoop);  }  //set the playing mode random  void Player::SetRandomPlayingMode(){  MediaPlayerlist.setPlaybackMode(QMediaPlaylist::Random);  } |

1. **播放中的异常处理**

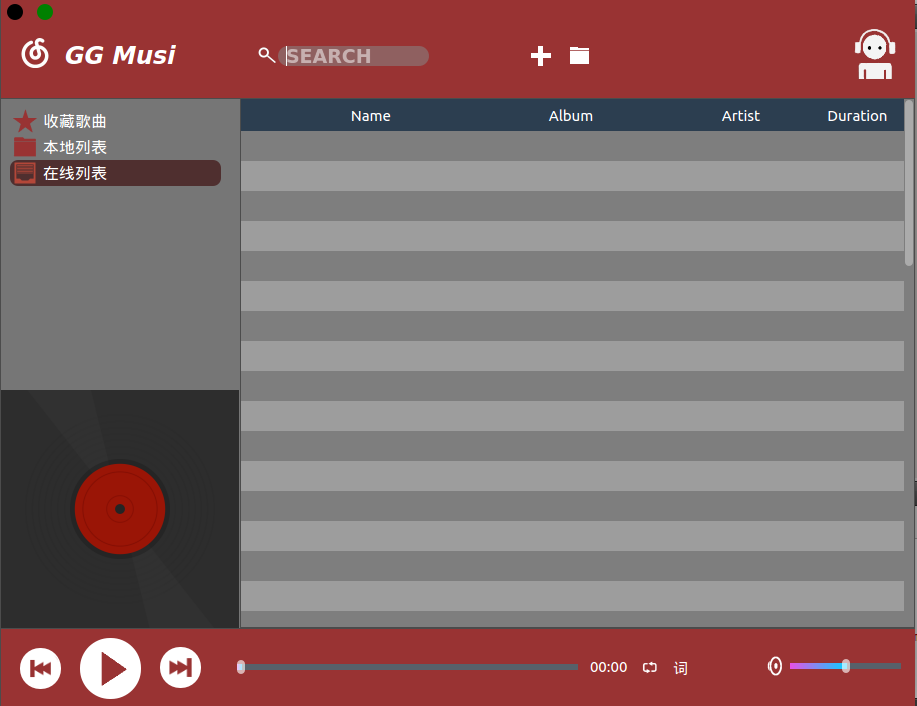
此播放器中主要设置了四个异常，分别为：

* AddToListException：添加到播放列表错误。添加单个文件或添加整个文件夹时的异常
* PlayNewMusicException：播放错误。主要是由于歌曲源文件不存在而播放的播放异常
* DeleteFromListException：删除错误。从播放列表中删除歌曲时的异常
* AlreadyInListExcption: 添加到播放列表错误。由于歌曲已经在列表中而重复添加

|  |
| --- |
| class AddToListException : public QException  {  public:  void *raise*() const {throw \*this;}  AddToListException \**clone*() const{  return new AddToListException(\*this);}  };  class playNewMusicException : public QException  {  void *raise*() const {throw \*this;}  playNewMusicException \**clone*() const{  return new playNewMusicException(\*this);}  };  class DeleteFromListException : public QException  {  void *raise*() const {throw \*this;}  DeleteFromListException \**clone*() const{  return new DeleteFromListException(\*this);}  };  class AlreadyInListException : public QException  {  void *raise*() const {throw \*this;}  AlreadyInListException \**clone*() const{  return new AlreadyInListException(\*this);}  }; |

1. **测试记录**
2. **播放器的初始化以及列表存储**

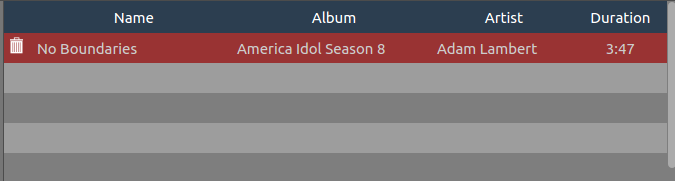
我们第一次运行此程序，初始化打开列表为空，图标为停止状态，音量初始化设置为50，这与我们initial函数中相一致。

****

**图二 播放器初始化界面**

1. **播放列表的增加和删除歌曲**

我们先通过图上的加号按钮添加一个文件，添加了一首歌名为No-Boundaries的歌，发现在列表中显示出来（见图三），此说明了我们的添加文件函数AddLocalMusic的正确性。然后我们点击图中的添加文件夹按钮，添加一整个文件夹中的歌曲，发现文件夹中的歌曲在列表中显示（见图四），表明添加文件夹AddLocalMusicFolder的正确性。我们注意到原来我们的歌单中已经存在了一首歌名为No-Boundaries的歌，后面我们添加的文件夹中也存在该首歌，而最后我们显示的只有一首。因为我们的添加功能会过滤到相同的歌曲，不会再重复添加。

****

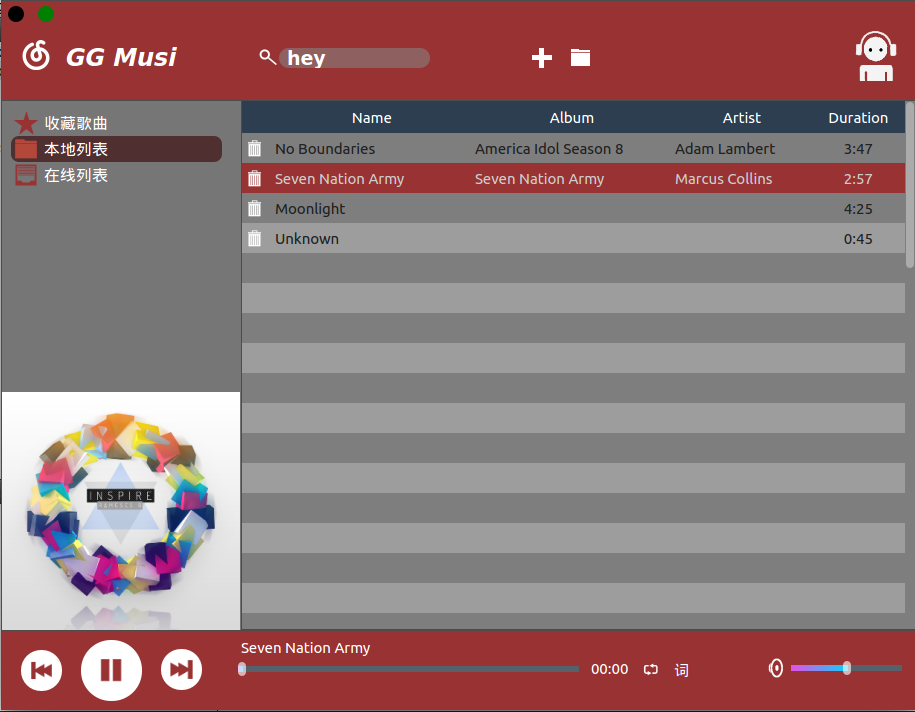
**图三 添加单首歌曲播放列表截图**

****

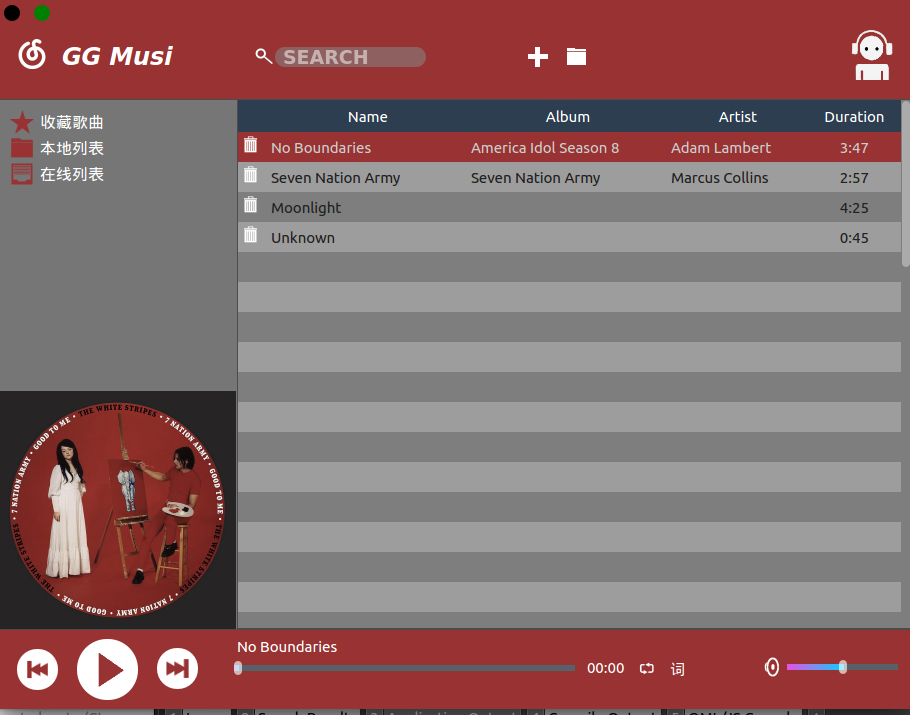
**图四 添加文件夹播放列表截图**

1. **播放器的控制功能实现**

我们双击歌曲Seven Nation Army，发现其能够正常的播放（见图五），并且进度条能够随着音乐的播放而变化，这说明了我们playNewMusic的功能的正常运行。我们点击中间的按钮按键发现其能够暂停并且再次点击其能够继续播放说明了我们的Retreat函数的正常运行。然后我们点击下一首，发现歌曲会调到下一首No Boundaries播放（见图六），这说明了我们的上（下）首功能的正常运行。

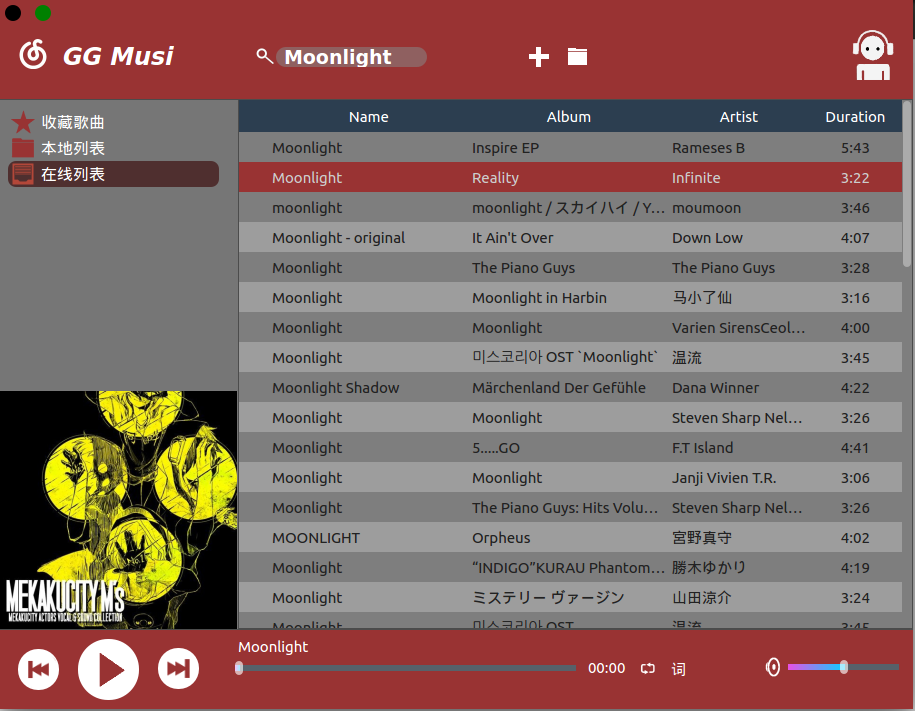
****

**图五 播放歌曲测试截图**



**图六 上（下）首功能测试截图**

我们连网后搜索出一些歌曲，双击其中的一首进行播放，发现其能够正常的播放（见图七），因而这说明了我们playOnlineMusic功能的正常运行。

****

**图七 在线音乐播放测试截图**

我们拖动界面上的进度条，发现其中歌曲的会调到我们拖到的相应的位置进行播放，并且我们拖动音量条，音量会随着我们的拖动而发生相应的变化。而关于播放模式的方面，我们分别设置了随机，单曲，顺序播放三种模式，我们改变三种播放模式，发现播放过程中播放会和我们的播放模式相适应，这表明了其正常运行。由于以上的部分都是动态变化的过程，截图不能显示，因而在此没有进行运行的截图，具体的运行在课堂演示上都有所体现。

1. **实验总结与分析**

此次实验我们通过MVVC模式小组合作开发了一个音乐播放器。通过此次的实验，我体会到了真正科研实践中是怎样进行合作开发的，了解了MVVC模式是怎样的并且明白了其在多人合作开发时的优点。此次的合作开发经历让我了解了在团队中合作的重要性，以及一个科学的开发模式在开发过程中会为后续的拼接带来很多的便利，以及为以后代码的修改也带来了极大的便利。此次开发要求我们使用GitHub，虽然以前也有所使用，但不是很熟悉，此次的开发经历让我学会了使用Git，体会了版本管理工具在多人开发的软件中会我们的开发效率带来极大的提高。此次开发经历学习到了真正科研实践中的一些知识，而这些知识是在平常的教学中是学习不到的，得到了很大的启发，也收获很多。