基于 QT 的音乐播放器的设计与实现

梁奇龙 白俊鸽
四川大学锦城学院 计算机与软件学院 四川 成都 610000

【摘 要】中国市面上大多音乐播放器存在许多不足之处，如内置广告，试听，后台占用内存大等问题。本文中的音乐播放器，内容简洁，可以很好的解决上述问题。开发通过 QT，C++ 技术，实现了歌曲的播放，快进，改变播放模式等功能，能够为客户提供良好的体验。

【关键词】音乐播放器；QT；C++

1 背景

如今我们的生活压力不断变大，人们越来越喜欢用听歌来缓解自身的压力，人们对于音乐播放器的要求变高，市面上出现了许多品种多样的播放器。同时，由于软件功能的多样化和复杂性，这些音乐播放器很难做到性能优异[1，2]。现在的播放器大多后台占用内存大，对于电脑性能有影响，市面上需要出现一个轻量级的，但是又能够满足人放松心情需求的小型播放器。

2 系统内容

播放器针对轻量级用户，为用户的音乐播放需求提供了简洁明了的操作系统支持。具体为：歌曲选择系统，播放系统，播放模式变更系统。

3 主要功能描述

对于一个轻量级的音乐播放器而言，在制作时不需要那些复杂繁琐的功能，只对于关键的部分给予保留，主要的功能有：歌曲清单，播放控制部分，音量控制部分，播放进度控制部分和播放模式控制部分。用户可以看到歌曲的名称，作者和存储模式，方便选择自己的歌曲，没有选择歌词的显示，因为可以节约软件所占的内存，方便后台挂起，而不会对用户电脑的性能产生影响。

4 界面设计

打开播放器，便可以看见该小型播放器的全部功能，在最上面是歌词的列表，显示在本地存储的歌曲，歌曲文件名进行过解析，每一行都只显示歌曲名，作者和歌曲类型，可以让使用者直观的看到能够播放的歌曲，让人很快的选择到自己喜欢的歌曲。其下方是播放进度条，在左右两侧有歌曲总时长和当前播放时长，可以让人方便观察播放进度，进度条也会随着播放时间进行移动，并且可以手动拉动，方便人进行快进。最下方是模式控制和上一曲，播放，下一曲，以及音量控制部分。播放模式有顺序，单曲，以及随机三种模式，可以满足市面上大部分人的需求。上一曲，播放，下一曲无需多言，音量控制既可以靠滑块拖动，也可以直接点击音量按钮来直接静音。界面总体比较简洁明了，占地小，但在必要功能上也是很完全的。

5 功能实现

5.1 歌曲显示

首先我们要把歌曲在界面上显示出来，歌曲是在本地的 D:\\music 目录下存储的，通过 musicPath 来存储这个目录，我们通过 设计 getfilename 这个函数来获取歌曲名，这个函数的参数是歌曲的存储地址，在函数内，我们创建 QDir 型的变量 dir 来存储这个地址，要使用 QDir 的变量，首先我们要在头文件中加入 #include<QDir> 这个头文件，这样我们才能够使用，我们要在目的地址查找的文件是 mp3 格式的，所以我们创建一个 QStringlist 型的变量 nameFilters 来存储 <<”*.mp3” 这个后缀，要使用 QStringlist 也需要在头文件中加 #include<QStringlist> 才能够使用，最后通过 QStringList files = dir.entryList(nameFilters，QDir::Files|QDir::Readable，QDir::Name)；这个语句来获取到音乐文件的文件名，这是一个标准的遍历文件的方式，最后我们返回 files 便可以得到音乐文件的文件名。然后得到了该怎么显示出来呢，我们在 QT 的 ui 界面选择 listWidget 这个 Widget 来显示歌曲名，将其命名为 listWidget，在文件初始化函数 init() 中，因为我们 getfilename 这个函数最后返回的是一行一行的歌曲名列表，所以我们用 QStringlist 型的 filelist 来调用 getfilename 这个函数以存储获取到的歌曲名，通过一个 for（int

i=0; i<filelist.size(); i++）循环来把歌曲名输出到界面上，QString filename = filelist.at(i)；获取到每一行歌曲名，最后通过 ui->listWiget->additem(filename) 将歌曲全部都输出到界面上。这样我们的界面上便可以完整的显示歌曲名了。

5.2 播放，上一曲与下一曲

然后我们要实现歌曲的播放以及上一曲，下一曲。音乐播放器的制作有着一个封装非常完整的类 QMediaPlayer，我们通过调用这个类可以让播放器的制作变得简单许多，而想要使用这个类的话，我们需要在头文件中加上 #include <QMediaPlaylist>，#include <QMediaPlayer> 这两个头文件，除此以外，还需要在 .pro 文件也就是 qt 的工程文件中加入 QT +=multimedia 这样一句。这样 QMediaPlayer 这个类就能够使用了。通过在头文件中定义 QMediaPlaylist *playlist； QMediaPlayer *player；然后在 .cpp 文件中 init() 函数里进行实例化，player =new QMediaPlayer； playlist = new QMediaPlaylist；这样就可以使用 player 和 playlist 了，在实现歌曲名显示的那个循环中，我们同时为 playlist 添加值，playlist->addMedia(QUrl::fromLocalFile(musicPath+"\\"+filename))；这样便把歌曲添加到播放列表中，然后通过 player->setPlaylist(playlist)；把播放列表添加到播放器中。这样进行播放以及上一曲，下一曲的准备工作就做好了。在 ui 界面分别设计三个按钮 buttonStart，buttonPrevious，buttonNext。首先我们来实现播放这个函数，播放这个按钮应该有播放和暂停两个功能，所以我们需要一个 bool playing 来判断当前的播放状态，通过 player->play() 来播放，player->pause() 来暂停，最后将槽函数进行连接即可，上一曲和下一曲的函数实现类似，上一曲通过 playlist->previous() 来改变播放位置，下一曲通过 playlist->next() 来改变播放位置。而这里需要关注的是边界的处理，我这里的设计是当在第一首点击上一曲时就还是进行第一曲的播放，通过 int current= =playlist->currentIndex()；来获取到播放位置，if(--current<0)，便把 current 置为 0，而在最后一曲点击下一曲时，就跳转到第一曲进行播放。通过 int row = playlist->mediaCount(); 来获取总的歌曲数，通过 int current = playlist->currentIndex()；来获取当前播放位置，如果 (++current>row)，就把当前播放位置为 0，然后进行 player->play()。最后进行槽函数的连接，把按钮的点击绑定到函数上，进行相应的响应。这样我们便实现了歌曲的播放，上一曲与下一曲。

5.3 音量控制

然后进行音量的控制，在 ui 界面生成一个按钮 buttonVolume 和一个滑块 verticalSlider，在 .cpp 文件中设计一个 SlotChangeVolume 的槽函数，参数为 int volume，player->setVolume(volume)；然后在槽函数的连接中将滑块的改变的值传给函数，这样便可以通过滑块的改变来控制音量。而按钮 buttonVolume 有静音和恢复音量的功能，与播放类似，需要判断，这里就不再多言，关键在于静音后如何恢复音量，所以需要对静音前的音量进行存储，oldvolume=ui->verticalSlider->value()；然后在恢复时将这个值传给 SlotChangeVolume() 函数就可以。然后进行槽函数的连接，把按钮的点击与函数进行绑定。这样便实现了音量控制。

5.4 进度条与播放进度时间

然后是进度条和播放时间与总时长，在 ui 界面设计一个滑块 playSlider 和两个标签 strTime，endTime。首先，我们需要把滑块的移动和歌曲的播放联系起来，我们设计一个槽函数 SlotsetPlayerPosition()，这个函数的参数是 int position，函数中使用 player->setPosition(position)；让播放的位置与滑块的位置相互联系，在槽函数的连接中使用 connect(ui->playSlider，SIGNAL(sliderMoved(int))，this，SLOT(SlotsetPlayerPosition(int)))；这样滑块的改变就可以把值传给函数，同时也能够通过手动拖动滑块来改变播放位置。然后需要设计函数 updatePosition（），参数为 qint64 position，也就是当前的滑块位置，这个函数的作用在于让滑块的移动与当前播放时间 strTime 进行联系，通过 currentTime =new QTime(0，(position/60000)%60，(position/1000)%60)；这里需要把头文件 #include<QTime> 加上，通过这个语句，得到当前时间的分和秒，这里没有考虑小时的情况，想的是应该没有这么长的歌。然后通过 ui->strTime->setText(currentTime->toString("mm:ss"))；就可以把当前时间显示出来，槽函数连接为 connect(player,&QMediaPlayer::positionChanged, this, &MainWindow::updatePosition)；把滑块位置的改变作为函数的响应。endTime 与这个类似，但笔者在实现 endTime 时同时也进行了滑块步长的处理，所以也讲一下，设计函数 updateDuration，参数为 qin64 duration，通过歌曲的变化导致总长度的变化。首先 ui->playSlider->setRange(0，duration)；这个语句来根据播放时长来设置滑块的范围，但是滑块的长度是固定的，所以我们只能改变滑块的步长 ui->playSlider->setPageStep(duration/10)；最后槽函数连接，用滑块长度的改变作为这个函数的响应，这样当滑块移动时不但当前时长会发生变化，当播放歌曲变化时，总时长也会发生相应的变化。

5.5 播放模式

然后是播放模式的控制，通过在 ui 设计一个按钮，

然后在 .cpp 文件中设计一个 SlotButtonPattern() 函数，因为有三种播放模式，这里的设计是点击一次便进行改变，所以要进行判断，用 playPattern 分别等于 0，1，2 来分别表示顺序播放，单曲循环，随机播放，通过函数内把在不同模式下 playPattern 的值进行控制，让每次点击都能够改变播放模式。通过 playlist->setPlaybackMode(QMediaPlaylist::Sequential)；来进行顺序播放，其余的播放模式类似。然后进行槽函数的连接，把按钮的点击与函数进行绑定即可。

5.6 播放歌曲高亮

然后是在播放时让播放歌曲的名字高亮，通过设计函数 updateList(int value)，其内容为 ui->listWidget->item(value)->setSelected(true)；这样便可以让 listWidget 被选定的行高亮，通过槽函数的连接 connect(playlist, &QMediaPlaylist::currentIndexChanged, this, &MainWindow::updateList);

这样当播放列表改变时响应这个函数，同时在初始化 init() 函数中加入 ui->listWidget->item(1)->setSelected(true)；这个语句，让 listWidget 最开始时第一首歌曲进行高亮，这样在播放的歌曲变化后总是让播放的歌曲高亮，可以很明显的看到现在正在播放的歌曲是什么。

5.7 双击播放歌曲

接下来的功能是双击歌名播放，设计槽函数 on_listWidget_doubleClicked(const QModelIndex &index)，在槽函数的连接中使用 SIGNAL(doubleClicked(QModelIndex))，将双击的这个位置传给函数，这里传的是 QModelIndex &index，不是 int 型的值，不能直接传给 playlist，所以在函数内，首先要 int currentIndex=index.row()-1；获取到 int 型的值，然后把 currentIndex 传给 playlist，而当双击歌曲时，不管我们是否正在进行播放，都要播放这个歌曲，所以把播放状态标识 playing=true；然后 player->play() 进行播放。这样便可以在双击歌名时播放歌曲。

5.8 细节处理

最后是对于这个播放器的一些细节处理，在初始化函数 init() 中，对于播放器的初始音量进行了初始化，player->setVolume(volume)；以防开始音量过大。对于每一个按钮，都为其设置了 ui，音量对于大小的不同，也有着不同的 ui。在播放模式处，对每一个播放模式都通过 ui->buttonPattern->setToolTip(" 顺序播放 ")；这样的进行了解释，可以通过将鼠标放置上去来显示模式名称，在进行了后面的这些处理后，一个简单的音乐播放器就制作完成了。

6 结束语

本文通过 QT，C++ 的模式进行音乐播放器的开发，通过引入 Qmediaplayer 这个类，让音乐播放器的开发变得简单了许多，开发成本低，代码简单易懂，使用 qt 进行项目开发，可以很好的实现程序的跨平台开发。虽然本文未实现对于一些复杂功能的实现，但是对于刚刚接触 qt 的初学者可以说是一个非常不错的锻炼项目。

【参考文献】

[1] 闫锋欣 , 曾泉人 , 张志强 .Blanchette J,Summerfield M.C++GUI Qt 4 编程 [M]. 北京 : 电子工业出版社 ,2013.

[2] 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.