程序设计实习

QT大作业报告

队名: 啊对对队
队长：汪轶寒
队员：吴桐 、范云天
2024年6月28日

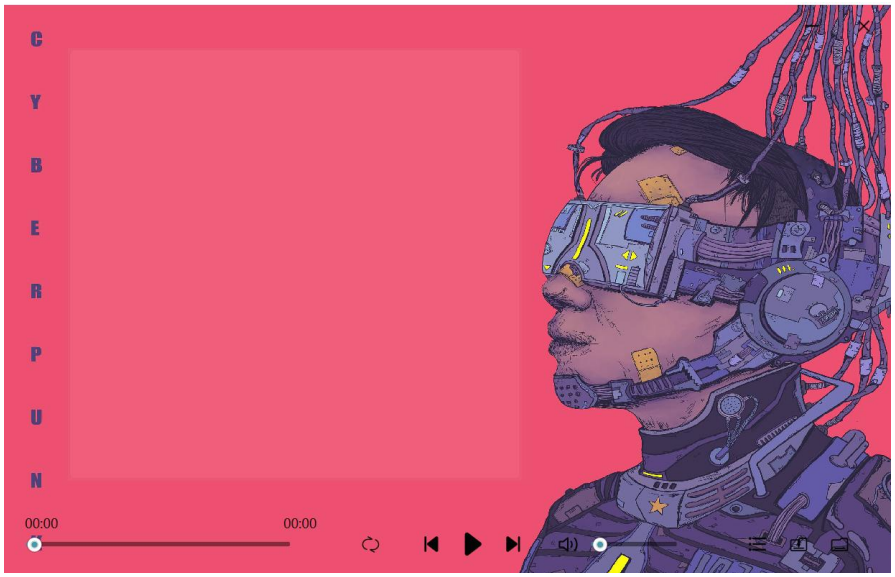
一．程序功能介绍

1. 项目名称：Music_Player （简易音乐播放器）
2. 基本功能：

（1）本地音乐文件获取及音乐下载：本地音乐文件获取支持.mp3 .wav .ogg等格式；也可以输入歌曲文件的URL实现音乐下载；

（2）基本播放功能实现：包括上一首、下一首、暂停、播放顺序调整、音量调整等。并实现了播放列表展示；

（3）播放器界面实现：播放器背景设置、进度条设置、播放列表字体优化。



二．项目各模块与类设计细节

项目主要包括音乐文件获取、音乐播放处理、播放器ui等模块，在头文件中定义有关槽函数，并在源文件中具体实现。

1. 音乐文件获取：

（1）本地音乐文件获取：定义槽函数 on_wenjian_clicked(),使用QFileDialog设置对话框属性，设置选择模式为目录选择；若用户在对话框中选择了一个目录并确定，获取用户选取的目录路径selectPath；使用QDir打开目录，通过entryList方法获取目录下扩展名为.mp3、.wav、.ogg的文件列表，并存储起来。若不为空，就将存储的音乐文件名添加到列表框中。

部分代码：

```
// 打开文件对话框，让用户选择音乐所在的目录
QFileDialog dialog(this);
dialog.setFileMode(QFileDialog::Directory); // 选择目录
dialog.setOption(QFileDialog::ReadOnly, false);  // 允许选择路径和文件
dialog.setDirectory(initialPath);
dialog.setWindowTitle("请选择音乐所在的目录");

// 显示文件对话框
if (dialog.exec()) {
    QStringList selectedFiles = dialog.selectedFiles();
    QString selectedPath = selectedFiles.first();
    qInfo() << selectedPath;

    // 根据路径，获取目录下的所有 mp3、wav、ogg 等音乐文件
    QDir dir(selectedPath);
    auto musicList = dir.entryList(QStringList() << "*.mp3" << "*.wav" << "*.ogg", QDir::Files);
    qInfo() << musicList;
```

（2）在线音乐下载：通过获取歌曲URL地址，下载音乐。项目中定义了DownloadUrlDialog类解决这一问题，DownloadUrlDialog继承自QDialog。这里使用CSS-like的语法定义对话框样式，并设置对话框为模态对话框，用户必须处理当前窗口才能处理父窗口和其他窗口。当ui中的okButton被点击时，调用accept槽函数,cancelButton被点击时，调用reject槽函数。Musicplayer.cpp文件中实现了on_wenjianadd()槽函数，获取URL并通过调用get()方法发送GET请求到指定的URL地址，触发网络请求获取指定资源。

2. 音乐播放处理

（1）上一首、下一首：使用简单的对播放文件编号的更改实现:

```
curPlayIndex=(curPlayIndex-1) % playList.size();
```

（2）播放、暂停按钮：用switch语句实现，通过播放状态判断，若为Playerstate,调用pause()方法暂停播放；若为PauseState,调用play()方法开始播放。

（3）播放进度显示：

```
//获取当前媒体的时长，通过信号关联获取总音乐时长
connect(mediaPlayer,&QMediaPlayer::durationChanged,this,[=](qint64 duration)
{
    ui->totallabel->setText(QString("%1:%2").arg(duration/1000/60,2,10,QChar('0')).arg(duration/1000%60,2,10,QChar('0')));
    //不足10的话补0，更美观
    ui->playCourseSlider->setRange(0,duration);    //让进度滑块随时间动起来(1)
});
//获取当前播放时长
connect(mediaPlayer,&QMediaPlayer::positionChanged,this,[=](qint64 pos)
{
    ui->curlabel->setText(QString("%1:%2").arg(pos/1000/60,2,10,QChar('0')).arg(pos/1000%60,2,10,QChar('0')));
    ui->playCourseSlider->setValue(pos);          //让进度滑块随时间动起来(2)
});
//拖动滑块改变播放进度
connect(ui->playCourseSlider,&QSlider::sliderMoved,mediaPlayer,&QMediaPlayer::setPosition);
audioOutput->setVolume(0.5);
```

a. mediaplayer媒体总时长变化时，触发durationChanged信号，获取当前媒体时长，设置滑块移动范围；

b. mediaplayer当前播放位置变化时，触发positionChanged信号，获取当前播放时长，设置滑块此时的位置；

c. 连接进度滑块的sliderMoved信号，移动滑块可以改变播放位置。

（4）音量调节：通过setVolume（）函数，可以移动音量滑块调整音量大小。

3. 播放器ui制作

（1）播放器背景设置：在MusicPlayer类中重写paintEvent函数，绘制音乐播放器背景图像；

```
void MusicPlayer::paintEvent(QPaintEvent *event)
{
    QPainter painter(this);//背景设置
    painter.drawPixmap(0,0,width(),height(),QPixmap(":/QTMusicPlayer/bk0.jpg"));
}
```

（2）优化播放列表字体，优化效果如下：

三．小组成员分工

小组按上面所述的项目模块进行分工，组长汪轶寒负责音乐文件获取部分，组员吴桐和范云天分别负责音乐播放处理和播放器ui。在具体实现过程中，主要按各个功能依次进行，先基本实现音乐文件获取部分的功能，再循序渐进实现各个按钮的具体效用。

四．项目总结与反思

项目在github上合作开发，切实感受到git在团队合作开发中的便捷。同时，在编写代码过程中，进一步加深了C++类和对象的知识，并学习了qt独特的信号与槽的机制，进一步理解了高内聚低耦合的设计原则。

在项目开发上，队伍也有诸多不足。如在时间安排上与预想有差距，部分功能尚未完善，或者尚未实现。希望在未来更多、更广阔的开发场景中，运用本次项目总结的经验，同时避免本次项目的不足之处。