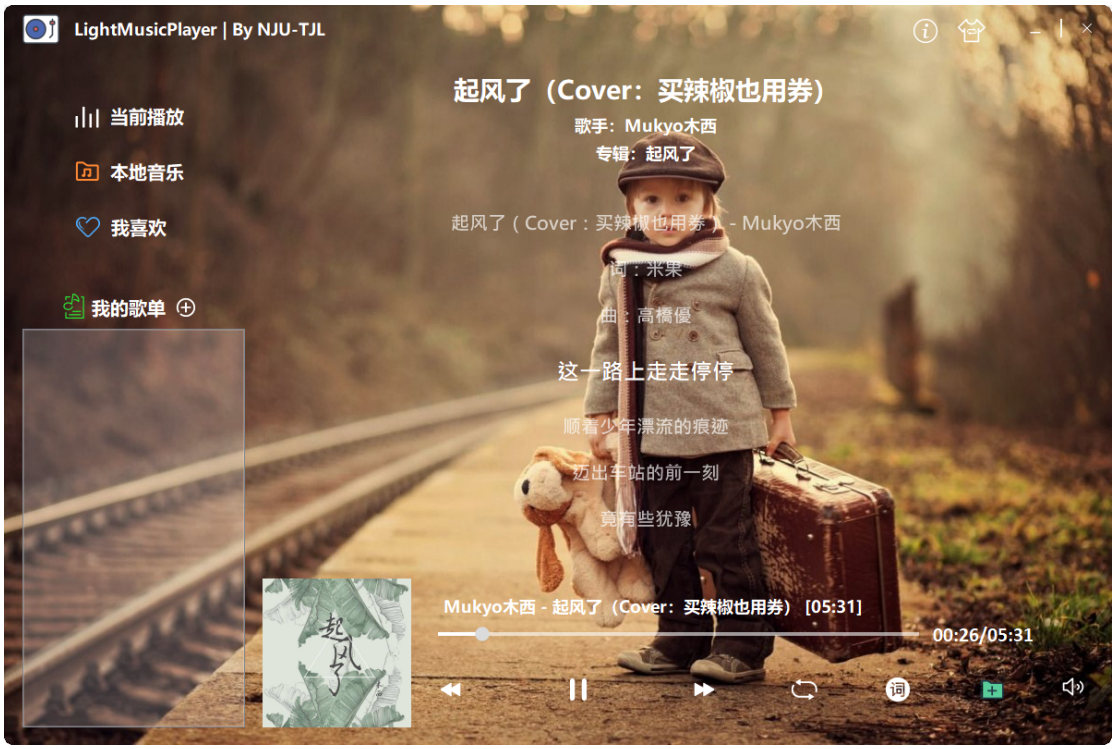
课程设计三

音乐播放器



姓名：唐金麟

联系方式：TangJinlin@smail.nju.edu.cn

目录

音乐播放器

一、概述

主要内容：

利用 Qt5 GUI 框架及其中的音频播放模块，完成了一个小巧、界面精美的本地音乐播放器。支持添加本地音乐、添加歌单、歌词查看、自定义背景、系统托盘控制播放等。同时，使用 SQLite 本地数据库保存已添加的音乐、歌单的信息，且使用.ini 文件记录应用设置信息。

已实现的目标：

- 界面精美且支持自定义背景
- 支持播放的音乐格式：.mp3 文件、.flac 文件、.mpga 文件
- 支持解析并展示歌词文件（.lrc 文件）
- 支持解析歌曲信息（专辑、比特率、缩略图等等）
- 支持歌单管理
- 系统托盘可控制播放
- 数据库保存已添加的歌曲和歌单信息

二、主要类的设计

1）依次介绍各个主要类的设计：

a）Music 类

代表一首歌曲。记录着歌曲相关信息，其中最为重要的是歌曲的 url 信息（即：歌曲文件所在的路径），播放的时候需要根据 url 来将歌曲文件加入播放器。还记录下的一些其他的歌曲信息，可见于下图。

部分数据成员及成员函数展示：

```
6   //歌曲类
7   class Music
8   {
9   private:
10      //歌曲资源地址
11      QUrl url;
12      //歌手
13      QString author;
14      //歌曲名
15      QString title;
16      //时长
17      qint64 duration;
18      //唱片集
19      QString albumTitle;
20      //比特率
21      int audioBitRate;
```

```
27  public:
28      Music(){}
29      Music(QUrl iurl);
30
31      //返回歌曲Url
32      QUrl getUrl() const {return url;}
33      //返回歌曲的信息
34      QString getInfo() const;
35      //弹窗显示歌曲信息
36      void detail();
37      //存入数据库
38      void insertSQL(const QString& name);
39      //根据文件名来获取歌词路径
40      QString getLyricFile();
```

b) MusicList 类

MusicList 类将 Music 类组织为一个一维结构，便于管理。它表示的是一个歌单，里面包着对歌曲列表进行管理的相关方法。

部分数据成员及成员函数展示：

```
//歌曲列表
class MusicList
{
    //歌单名
    QString name;
    //所存储的歌曲
    vector<Music> music;
```

```
public:
    MusicList(){}
    MusicList(const QList<QUrl>& urls,QString iname="");
    //设定歌单名
    void setName(const QString& iname){name=iname;}
    QString getName(){ return name; }
    //设定数据库标志
    void setSQL(bool on){ sql_flag=on; }
    //从url添加歌曲
    void addMusic(const QList<QUrl>& urls);
    //添加一首歌曲
    void addMusic(const Music& iMusic);
    //获取指定位置的歌曲
    Music getMusic(int pos);
    //将本歌单加入播放列表
    void addToPlayList(QMediaPlaylist *playlist);
    //歌单可视化
    void addToListWidget(MusicListWidget *listWidget);
```

```
    //移除指定的歌曲
    void removeMusic(int pos);
    //在文件夹中打开
    void showInExplorer(int pos);
    //显示指定歌曲详细信息
    void detail(int pos);
    //数据库中移除全部本歌单的歌曲
    void remove_SQL_all();
    //将歌单中的歌曲全部写入数据库
    void insert_SQL_all();
    //从数据中恢复本歌单
    void read_fromSQL();
    //将本列表中的歌曲排序
    void sort_by(COMPARE key);
    //整理歌单：将歌单中的重复歌曲去掉并排序
    void neaten();
    //清空本歌单
    void clear();
};
```

c) MusicListWidget 类

以列表形式展示歌曲时，会用到 Qt 中的 QListWidget 组件，而直接使用 QListWidget 组件因为没有结合 MusicList 这个数据结构，所以不太方便管理。因此，MusicListWidget 继承了 QListWidget，它在类中包含 MusicList 类，这样可以将 MusicList 中的歌曲信息方便地可视化展示出来，并且可以方便地对展示出来的歌曲进行管理。

部分数据成员及成员函数展示：

```
    //当前歌曲列表（存储的是歌曲信息）
    MusicList musicList;
    //当前展示列表项使用的图标
    QIcon icon=QIcon(":/image/image/image/music.png");
public:
    MusicListWidget(QWidget *parent = Q_NULLPTR);
    //刷新显示（当musicList有所变化时，需要调用）
    void refresh();
    //设置歌曲列表
    void setMusicList(const MusicList& music);
    //设置播放列表
    void setMusicList_playing(const MusicList& music);
    //移除歌曲
    void removeMusic();
    //在文件夹中打开
    void showInExplorer();
    //歌曲详情
    void detail();
    //设置/获取图标Icon
    void setIcon(QIcon iicon){ icon=iicon; }
```

d) MusicListDialog 类

继承于 QDialog 类，在添加歌曲到“我喜欢”和歌单时，用于选择歌曲。这里使用自定义对话框主要为了：①便于独立使用一个.ui 文件进行界面设计；②便于获取歌曲数据（即本地音乐有哪些歌曲），以及返回选择结果。

部分数据成员及成员函数展示：

```cpp
class MusicListDialog : public QDialog
{
    Q_OBJECT
    //歌曲数量
    int num;
    //存放歌曲条目是否被选择（用于返回选择结果）
    bool *selected_flag;
public:
    explicit MusicListDialog(QWidget *parent = nullptr);
    ~MusicListDialog();

    //设定对话框展示出的歌单以及选择结果存放的地方
    void setMusicList(MusicList& ilist, bool *results);
```

e）LyricWidget 类

继承 QWidget，一个矩形区域，包含有用于显示歌词的几个文本框（QLabel），用于显示歌词。这里使用自定义的 Widget，也是便于独立使用一个.ui 文件进行界面设计。同时这样做，而不是在主窗口中直接显示歌词的文本框。也使得主窗口组件模块化，更加有条理。此类的主要功能就是解析歌词文件，且按播放进度显示。

部分数据成员及成员函数展示：

```cpp
class LyricWidget : public QWidget
{
    Q_OBJECT

    //储存所有歌词
    vector<LyricLine> lines;
public:
    explicit LyricWidget(QWidget *parent = nullptr);
    ~LyricWidget();

    //将歌词文件的内容处理为歌词结构的QList
    bool process(QString filePath);
    //根据时间找到对应位置的歌词
    int getIndex(qint64 position);
    //显示当前播放进度的歌词
    void show(qint64 position);
    //根据下标获得歌词内容
    QString getLyricText(int index);
    //清空歌词Label
    void clear();
```

f）MainWidget 类

主窗口，用户直接交互的一个窗口组件。负责初始化、处理播放器相关事件、响应用户交互事件（点击、拖动等）等等工作。主要使用.ui 文件在 Qt 设计师中完成界面设计。

部分数据成员及成员函数展示：

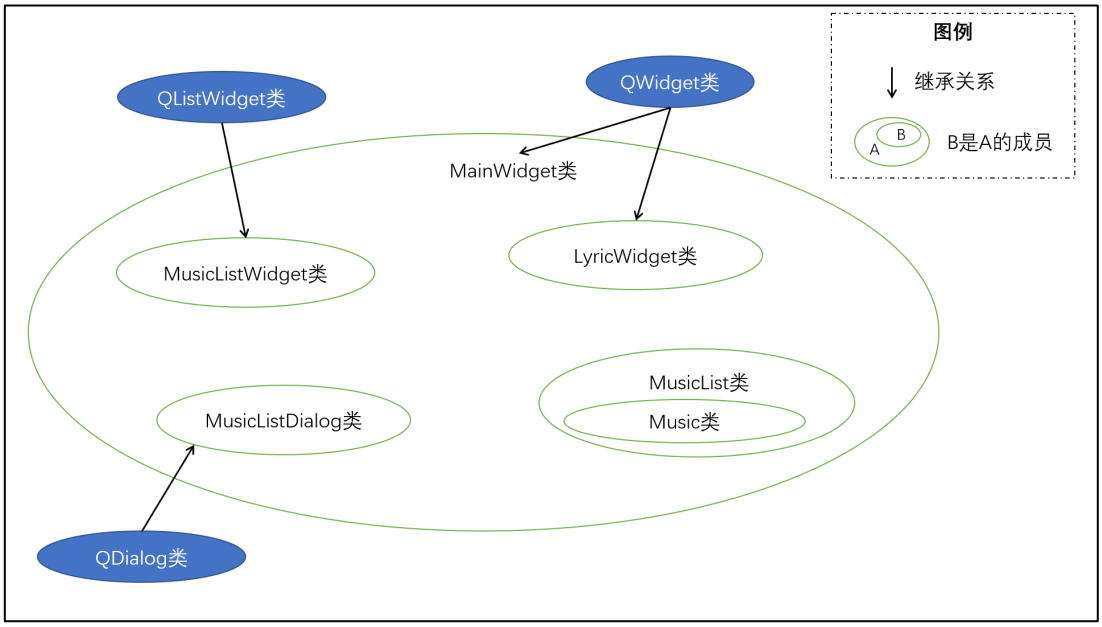
```cpp
class MainWidget : public QWidget
{
    Q_OBJECT

public:
    explicit MainWidget(QWidget *parent = nullptr);
    ~MainWidget() override;

private:
    Ui::Widget *ui;
    void paintEvent(QPaintEvent *event) override;

    //UI组件额外的一些处理
    void init_UI();

    //当前播放器
    QMediaPlayer *player;
    //当前播放列表
    QMediaPlaylist *playlist;
    //初始化一些成员变量以及connect连接
    void init_play();
    //更新播放信息相关（进度、信息等）
    void updatePosition(qint64 position);
    void updateDuration(qint64 duration);
    void setPosition(int position);
    void updateInfo();
    void updatePlayBtn();
```

```cpp
    //系统托盘
    QSystemTrayIcon *mySystemTray;
    QAction *action_systemTray_play;
    QAction *action_systemTray_playmode;
    //响应系统托盘的动作（双击等）
    void systemTrayIcon_activated(QSystemTrayIcon::Ac
    //退出应用
    void quitMusicPlayer();
    //系统托盘初始化
    void init_systemTrayIcon();

    //数据库初始化
    void init_sqlite();
    //配置初始化（配置文件读取）
    void init_settings();
    //“本地音乐”、“我喜欢”等歌单的初始化
    void init_musicList();

    //歌单
    vector<MusicList> musiclist;
    //用于标识现在展示的是哪个歌单
    int musiclist_index=-1;
    //更新展示歌单名字的listwidget
```

```
    //用于更新展示歌单内容的listwidget
    void musicListWidget_refresh();

    /*右键菜单*/
    //菜单项的初始化
    void init_actions();
    //“当前播放”列表的右键菜单
    QMenu *menu_playlist;
    //“本地音乐”列表的右键菜单
    QMenu *menu_locallist;
    //“我喜欢”列表的右键菜单
    QMenu *menu_favorlist;
    //“歌单名列表”的右键菜单
    QMenu *menu_namelist;
    //“歌单展示列表”的右键菜单
    QMenu *menu_musiclist;
    //更换皮肤的菜单
    QMenu *menu_changeSkin;
```

```
protected:
    //窗口拖动时记录的起始点
    QPoint offset;
    /*重写Widget的一些方法*/
    //实现窗口可拖动
    void mousePressEvent(QMouseEvent *event) override;
    void mouseMoveEvent(QMouseEvent *event) override;
    void mouseReleaseEvent(QMouseEvent *event) override;
    //关闭时不退出，而是到系统托盘
    void closeEvent(QCloseEvent *event) override;
    //拖拽文件进入
    void dragEnterEvent(QDragEnterEvent* event) override;
    void dropEvent(QDropEvent* event) override;

private slots:
    /*部分右键菜单项对应的操作（即对应QAction连接的槽函数）*/
    void playlist_removeMusic();//当前播放列表-右键菜单 移除歌曲
    void play_to_favor();//从当前播放添加到我喜欢
    void local_to_favor();//从本地音乐添加到我喜欢
    void local_to_playlist();//从本地音乐添加到当前播放列表
    void favor_to_playlist();//从我喜欢添加到当前播放列表
    void namelist_delete();//移除歌单
    void musiclist_removeMusic();//从歌单展示列表移除歌曲
    void musiclist_to_favor();//从当前歌单添加到我喜欢
    void musiclist_to_playlist();//从当前歌单添加到正在播放
    void background_to_default();//换到默认背景
    void background_setting();//自定义背景

    /*一些点击事件的响应（使用.ui中的部件“转到槽”自动生成）*/
    void on_btnCurMusic_clicked();
    void on_btnLocalMusic_clicked();
```

2）综上所述，各个类之间的关系大致如下图所示：



三、程序的功能特点和运行操作方法

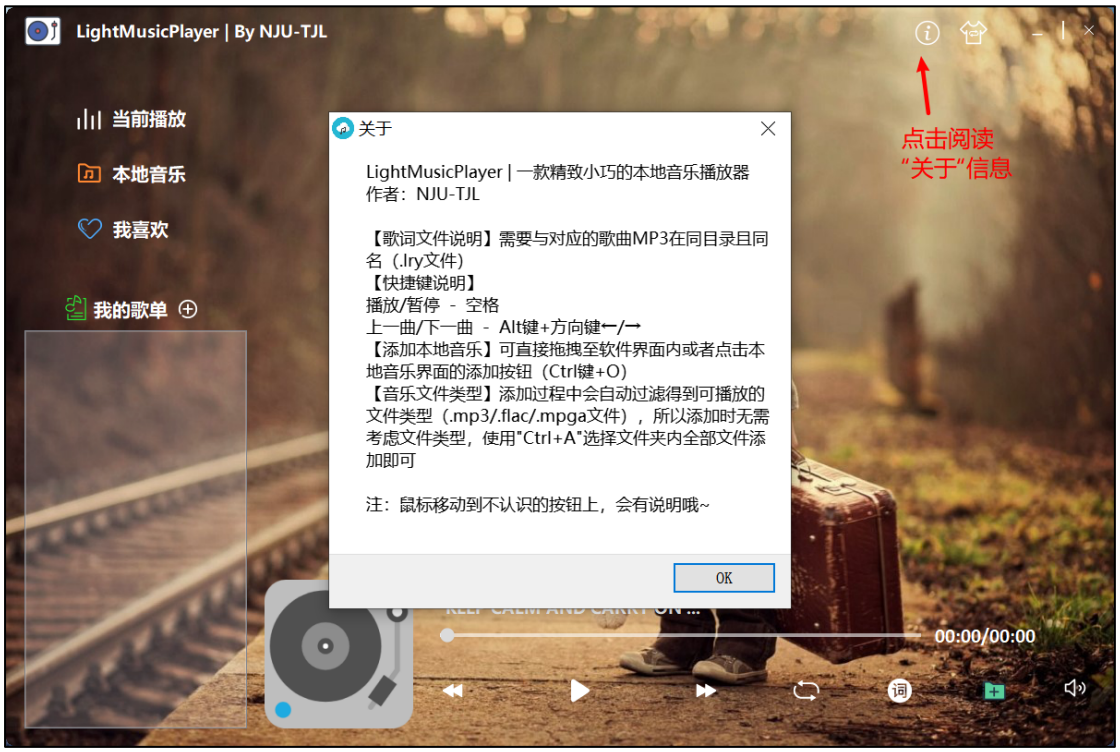
〈关于可执行程序的一些说明〉

① 可执行程序为单个.exe 文件。使用“Desktop Qt 5.9.8 MinGW 32bit”构建套件在 Release 模式下生成，然后使用 Enigma Virtual Box 软件将动态 DLL 库文件及其他相关文件打包生成了单个的一个.exe 文件。

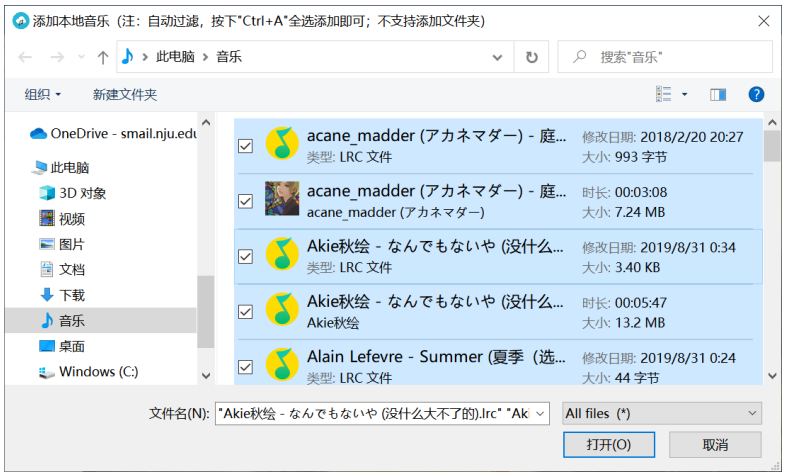
② 第一次运行后，会在同目录下产生两个文件：a)Music.db 文件。SQLite 本地数据库文件，存储已添加到播放器的歌曲的信息；b)LightMusicPlayer.ini 文件。保存用户设置信息，即如果

设置了自定义背景图片，会在此配置文件中记录下图片文件的路径。

双击运行即可打开。打开后，可通过点击左上角的图标或者右上方的“关于”按钮，弹出本软件的“关于”信息，其中包含一些说明和操作方法。



点击右下方绿色的按钮（或者使用快捷键“Ctrl+O”），可添加本地音乐。在弹出的窗口中，进入存放音乐文件的目录下，按下“Ctrl+A”全选，再点击打开即可完成添加（此过程中会自动过滤掉不支持播放的文件类型，所以此时用户不用手动筛选添加文件的类型）。



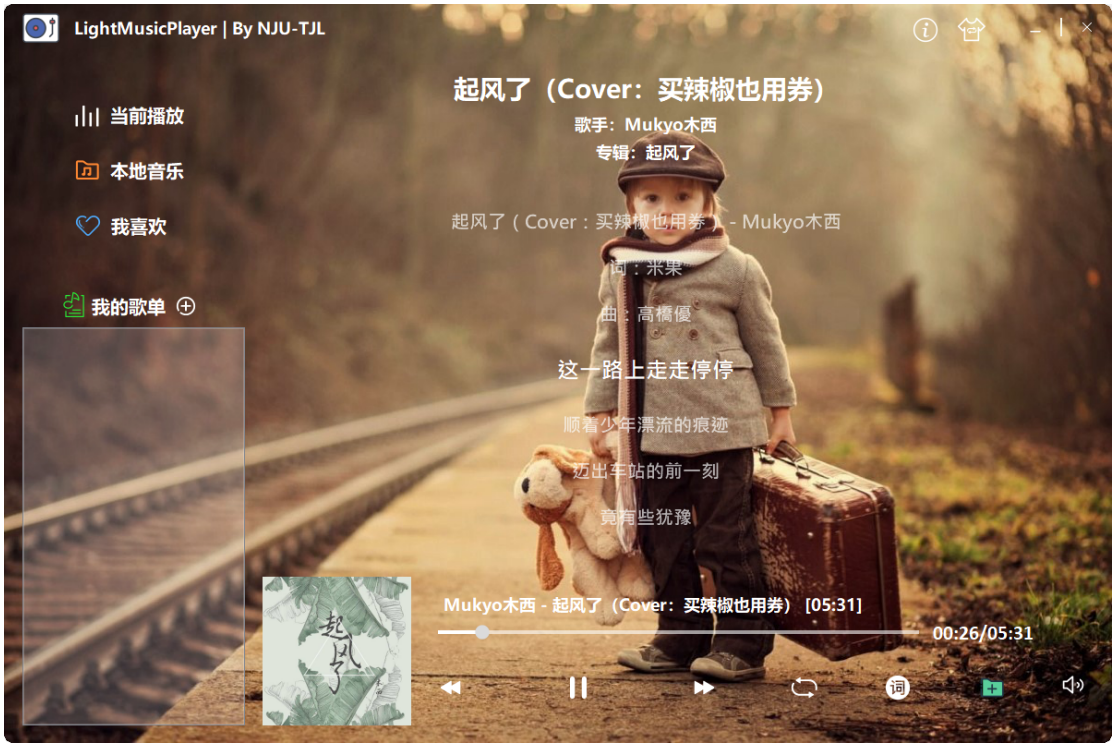
添加本地音乐操作，也可以直接拖拽文件至主窗口界面来完成，也会自动筛选文件类型。

从上图也可看出，播放器主窗口有四个的圆角（值为 10px），界面更加和谐。

添加后，本地音乐列表显示已添加的歌曲。上方的按钮分别表示：清空、整理歌曲（排序且去除重复）、按歌手排序、按歌曲名排序、按时长排序。

此时，双击某首歌曲，即可添加到当前播放，并且开始播放。下方会显示歌曲信息及歌曲图片。同时，当前播放列表中正在播放的歌曲的图标会有所不同。

点击下方的歌词按钮，可进入歌词显示界面。

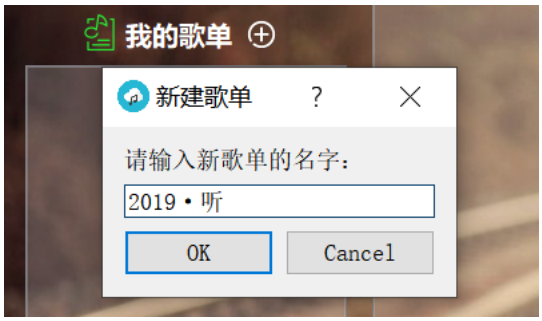


在本地音乐和当前播放列表中，右键点击某歌曲，可以弹出右键菜单。

点击我喜欢左侧的加号，可批量从“本地音乐”中添加歌曲。弹出如下的对话框。

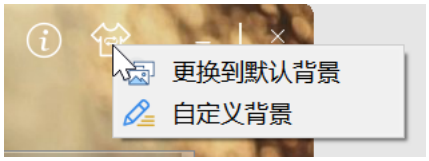
添加完成后，如下。(同样也支持右键菜单)

点击加号，新建歌单。同时，右键可删除歌单。



双击歌单名字即可进入对应的歌单详情列表。其中的操作与“我喜欢”列表类似。

右上角按钮，可更换背景图片、最小化，以及关闭主窗口。



主窗口关闭后，并不会退出播放器，而是最小化到了系统托盘。系统托盘实时显示正在播放的歌曲，且可通过右键菜单控制播放。

四、实现中值得一提的地方

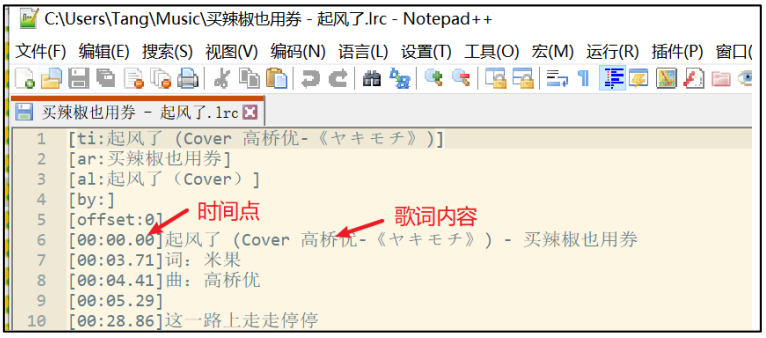
1）歌曲图片

平时从 QQ 音乐等播放器上下载的 MP3 歌曲文件，在 Windows 资源管理器中查看的时候，会有图片显示（专辑图）。所以一开始猜想，歌曲的图片信息也是存储在 MP3 文件中，那 Qt 的库函数可以解析出来吗？最终，查阅手册和根据这个 回答，找到了方法。只需要在解析歌曲文件时候，获取名为" ThumbnailImage"（缩略图）的元数据即可。

```
//封面图片（应获取"ThumbnailImage" From: https://www.zhihu.com/question/36859497）
QImage picImage= player->metaData(QStringLiteral("ThumbnailImage")).value<QImage>();
if(picImage.isNull()) picImage=QImage(":/image/image/image/non-music.png");
```

2）歌词展示

这里使用了 Qt 中的正则表达式来解析标准的 LRC 歌词文件，主要是需要识别出歌词的时间点，如下图。



然后在播放的时候，根据播放进度，比较时间点的先后，在对应的 QLabel 上展示出相应位置的歌词。

3）在 Qt Designer 中使用自定义的部件

主窗口的界面设计主要使用.ui 文件完成，而主窗口中的部分组件使用的是自定义的部件（比如 MusicListWidget 和 LyricWidget），但是 Qt Designer 左侧可拖拽出的部件都是 Qt 中的自带部件，那么如何使用自定义部件在 Qt Designer 中完成设计呢？

这里可以使用"部件提升"功能，在右侧的"对象与类"窗口，右键，选择"提升为"，即可将某个 Qt 中的标准部件，修改为使用自定义的部件（该部件必须继承自对应的标准部件）。

4）界面设计总结

① 字体使用"微软雅黑"的粗体比较美观；
② 方框背景设定为白色，且设置一定的透明度；
③ 善用.ui 文件同时结合 QSS 样式表进行设计，方便控制间距、位置等；

④ 图标的选择、颜色的使用、摆放位置等，注意是否和谐;
⑤ 图标的挑选可使用 阿里巴巴矢量图标库 。

全文结束，感谢阅读！Thanks For your time...