**Python大作业报告**



**系统名称： 音乐播放器**

**姓 名： 杜瑜琛**

**班 级： 计科2001**

**完成日期： 2023/1/7**

**成绩评定表**

|  |  |  |
| --- | --- | --- |
| **成绩** | **教师签名** | **评定时间** |
|  |  |  |

目录

[**Python大作业报告** 1](#_Toc124001237)

[**成绩评定表** 1](#_Toc124001238)

[**成绩** 1](#_Toc124001239)

[**教师签名** 1](#_Toc124001240)

[**评定时间** 1](#_Toc124001241)

[一、 背景介绍 1](#_Toc124001242)

[二、 功能介绍 2](#_Toc124001243)

[1.导入本地歌曲 2](#_Toc124001244)

[1.1导入指定文件夹的音频文件 2](#_Toc124001245)

[1.2导入指定磁盘文件音频文件 2](#_Toc124001246)

[2.播放音乐 2](#_Toc124001247)

[3.自定义歌单 2](#_Toc124001248)

[3.1 ‘like’歌单 2](#_Toc124001249)

[3.2 其他歌单 2](#_Toc124001250)

[3.3 歌单字体和大小 2](#_Toc124001251)

[三、 详细使用说明 3](#_Toc124001252)

[1. 启动软件 3](#_Toc124001253)

[2. 本地音乐 4](#_Toc124001254)

[3. 播放列表 5](#_Toc124001255)

[4. 歌单 6](#_Toc124001256)

[5.播放切换歌曲功能 6](#_Toc124001257)

[四、 总结 6](#_Toc124001258)

[附：系统源代码 7](#_Toc124001259)

[1、qt.py 7](#_Toc124001260)

[2、json文件 30](#_Toc124001261)

# 背景介绍

随现今社会生活紧张，而欣赏音乐史其中最好的舒缓压力的方式之一，音乐播放类的软 件数不胜数，为什么我还要再写一个播放器出来呢？因为现有的音乐播放器功能实在是有些多了，多未必不好，但是我总想要一个干净纯粹的音乐播放器，只为听歌的播放器，一个有着一些自己想要的功能的播放器，比如“切歌”，一个自己熟悉代码的播放器，甚至，一个可以自定义功能结构的音乐播放器。本项目的目 的是开发一个可以播放主流音乐文件格式的播放器，本设计的实现的这主要功能是播放MP3等音乐文件，并且能够控制播放器播放，暂停，停止，上一曲，下一曲。界面简单，操作简单。 本软件具备音乐播放器的播放歌曲、歌曲列表、拖动进度等基本功能，同时本播放器界面简洁美观，操作简单便捷。

本系统开发采用了pyqt5，json工具，用pycharm编写代码

# 功能介绍

1.导入本地歌曲

1.1导入指定文件夹的音频文件

选择相应路径，点击选择文件夹，系统会将当前文件夹的所有音频文件处理好放在播放列表中，包括歌曲的歌曲名、歌手、专辑、时长等信息。

1.2导入指定磁盘文件音频文件

与1.1导入指定文件夹的音频文件类似，但是不需要选择文件夹，而是选择直接询问除c盘外的所有磁盘文件，将选中的磁盘文件中所有的音频文件导入播放列表中。

2.播放音乐

在播放列表中，这里有三种方式实现，一个是单击选中歌曲，窗口底部的“播放”按钮；

另一个是右键的下拉菜单中选择播放；或者双击直接播放歌曲。

设置了音量调节，可以拖动进度条实现音量改变。

设置了音乐播放的进度条，左侧是已播放的时间，右侧是此音乐的总时长，可以进行拖动调整进度。

底部的按钮有上一曲下一曲，进行歌曲的轮换

设置了播放模式，包括随机播放、列表循环、顺序播放、单曲循环

3.自定义歌单

3.1 ‘like’歌单

初始就存在的like歌单存放喜欢的音乐，有☆/★标记的歌单，会在播放列表中显示是否已收藏进like歌单，不可删除

3.2 其他歌单

初始没有歌单，需要创建新的歌单，可以通过右键音乐加入歌单，歌单可以删除

3.3 歌单字体和大小

在新建歌单上方可以通过下拉菜单调整字体和大小

# 详细使用说明

## 启动软件

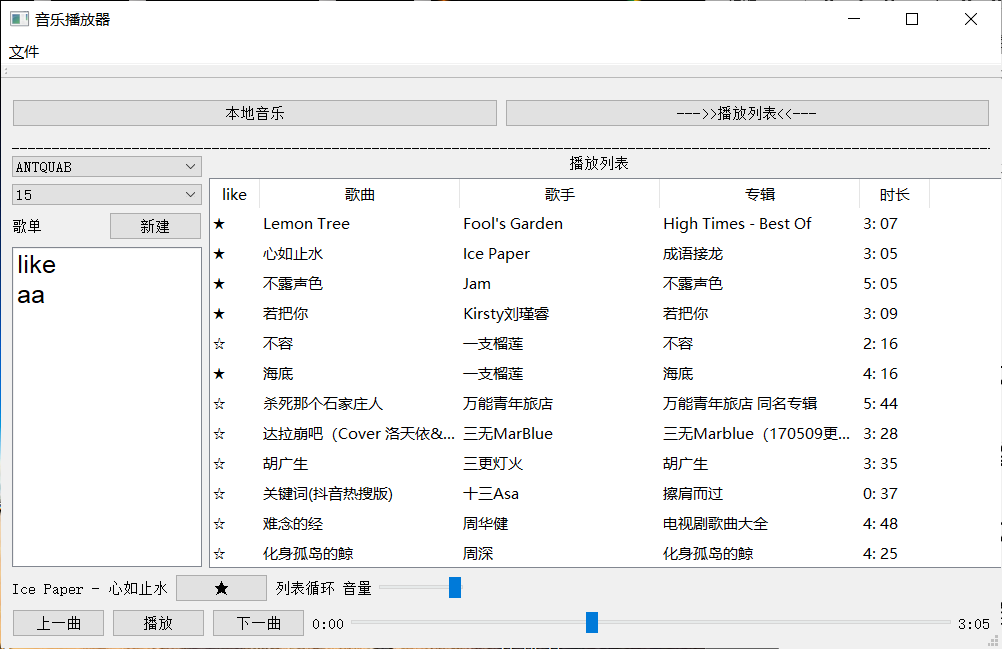


图3.1主页面

## 2. 本地音乐

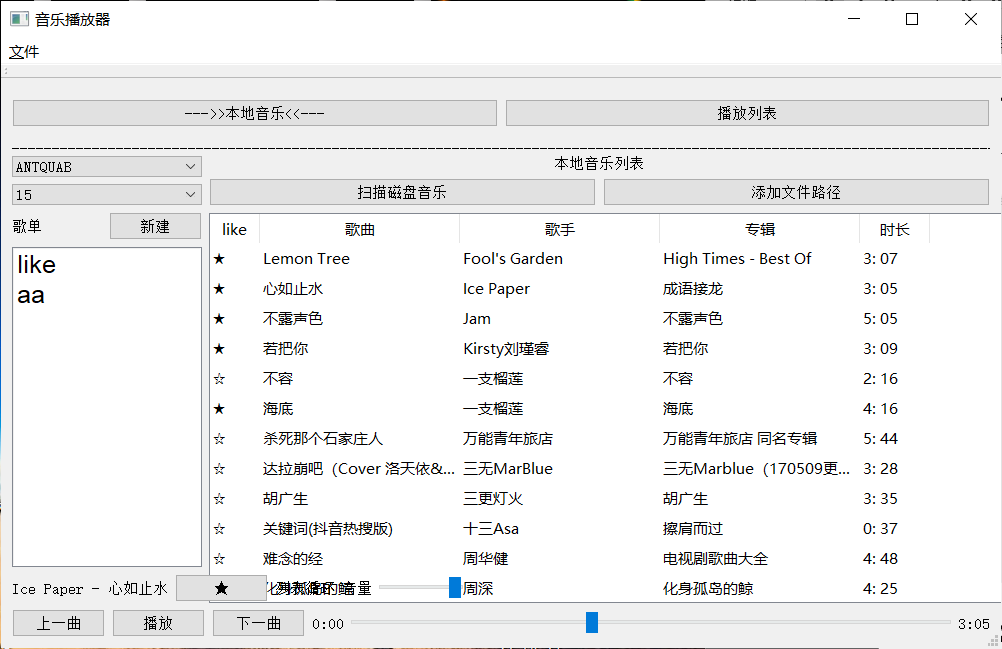


图3.2本地音乐\_0

点击扫描磁盘音乐可得到图3.3,可选择相应路径扫描当前文件夹的音频文件

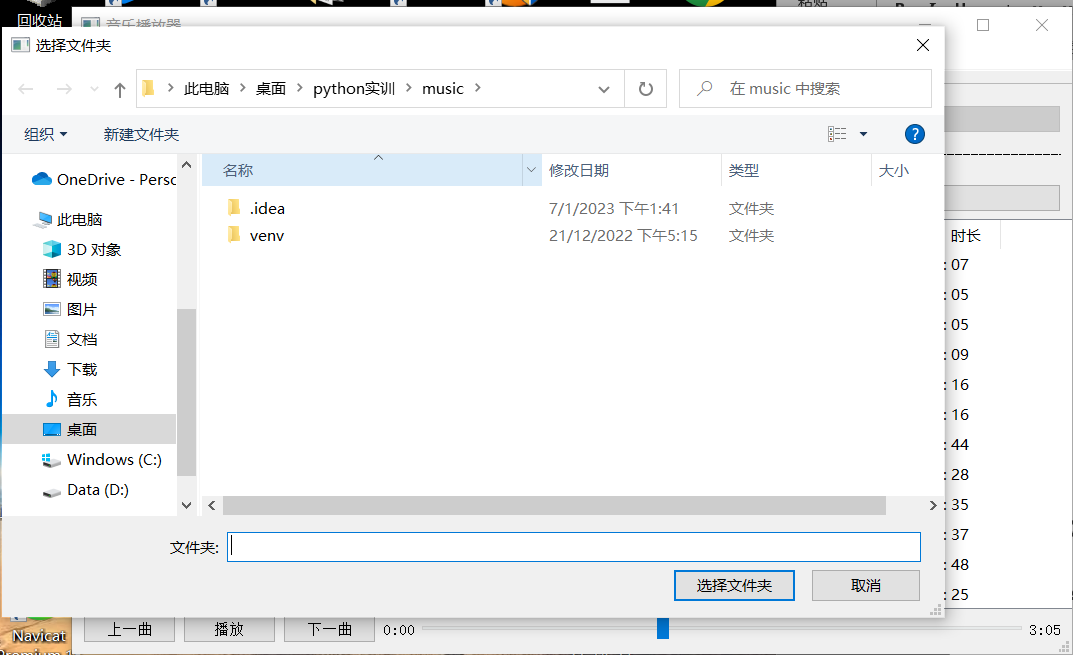


图3.3本地音乐\_1

点击图3.2扫描磁盘音乐可选择相应的磁盘进行磁盘音乐文件扫描

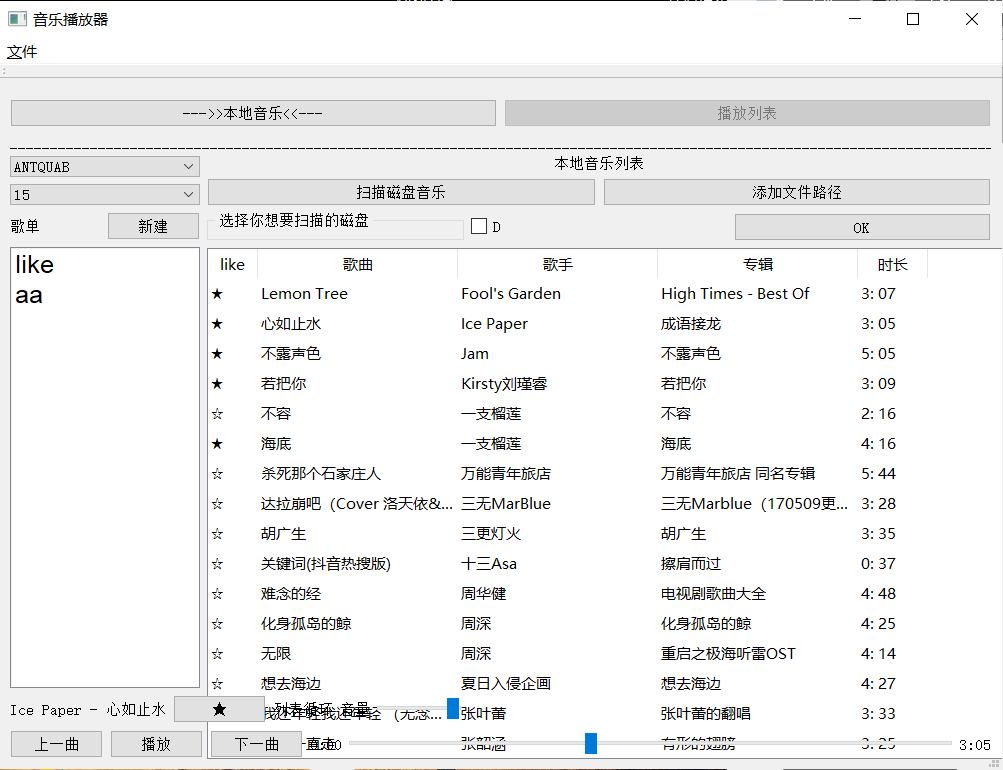


图3.4本地音乐\_2

由以上方法扫描到的文件获得的音乐会放在本地音乐列表中

## 3. 播放列表

播放列表的页面就是系统初始页面如图3.1，右键列表中的某一首音乐可以得到下拉功能选择

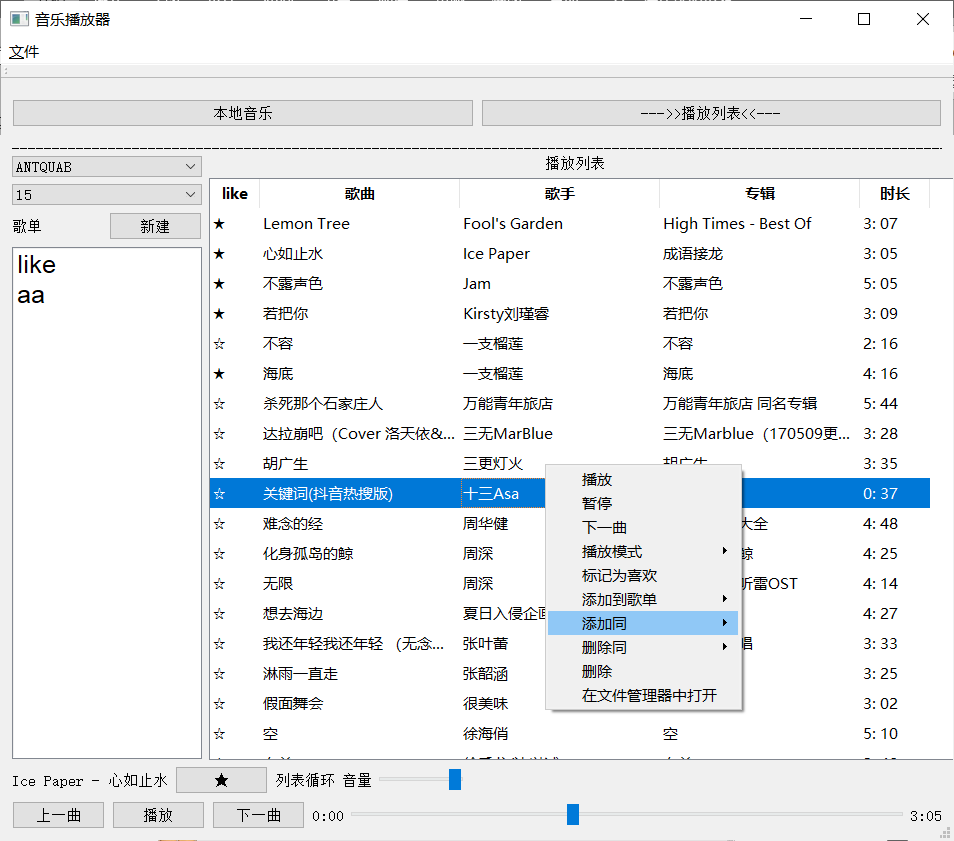


图3.5播放列表\_0

## 歌单

初始存在like歌单，也可以点击新建或者右键新建，输入想要的歌单名称点击OK即可，也可以

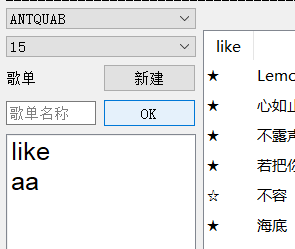


图4.1

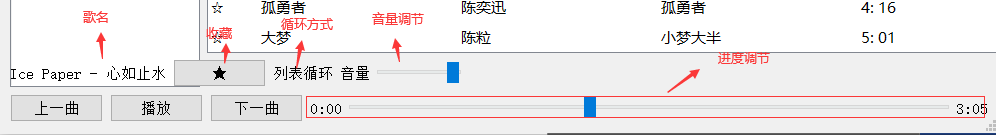
5.播放切换歌曲功能

图5.1音乐播放功能

# 总结

通过本次课程设计我学到了很多知识。首先，开始选题时比较迷茫，因为从来没

有独立完成过一个系统的开发，所以有很长时间都在思考自己该做，能做什么。后来

确定课题后又无从着手，不知到该怎么进行。在开发过程中又遇到很多不懂的技术，

查阅了很多的文献。在不断的解决问题中我取得了很大的进步。在老师细心指导下，

在同学的无私帮助下终于顺利完成了课程设计。

此次的毕业设计达到了既定的功能要求，也使我对多媒体音频方面的技术及理论

知识有了更深刻的理解。通过开发平台的使用，使我能够更加熟练的使用pycharm，与pyqt5

由于时间有限，本系统还存在- -些缺陷和不足。例如播放界面的设计等，这些待以后

有时问慢慢改进。但其功能齐全、占用系统资源少、运行平台要求不高、界面美观舒适、操作简单、易维护、高稳定性是本软件所特有

# 附：系统源代码

1、qt.py

该文件主要实现了所有的界面功能，具体代码如下：

**import json # time,**

**import numpy**

**import os**

**# from PyQt5.QtMultimedia import \***

**import pathlib as pa**

**import psutil**

**import sys**

**import tinytag**

**from PyQt5 import sip**

**from PyQt5.QtCore import Qt, QTimer**

**from PyQt5.QtGui import QFont, QCursor**

**from PyQt5.QtWidgets import \***

**from pygame import mixer**

**class All():**

**def \_\_init\_\_(self, all):**

**self.like = all['like'] # list**

**self.playlist = all['playlist'] # {}**

**self.musiclist = all['musiclist']**

**self.playingmusic = all['playingmusic']**

**self.playingtime = all['playingtime']**

**self.musicsec = all['musicsec']**

**self.playmode = all['playmode']**

**self.playhistory = all['playhistory']**

**self.allmusic = all['allmusic']**

**self.volume = all['volume']**

**# all={'like':[],'allmusic':[],'playlist':{},'musiclist':[], 'playingmusic':'', 'playmode':'随机播放', 'playingtime': 0, 'playhistory': {}, 'musicsec': 0,'volume'=1.0}**

**class ttt(QWidget):**

**def \_\_init\_\_(self, parent):**

**super().\_\_init\_\_(parent)**

**mixer.init()**

**self.initt()**

**def initt(self):**

**self.layv = QVBoxLayout()**

**self.layv\_h1 = QHBoxLayout() # top**

**self.layv\_h2 = QHBoxLayout() # bottom**

**self.layv\_h2\_v1 = QVBoxLayout() # bottom left 歌单 总**

**self.layv\_h2\_v1\_h1 = QHBoxLayout() # 歌单名称**

**self.layv\_h2\_v1\_h2 = QHBoxLayout() # 歌单创建**

**self.layv\_h2\_v1\_h3 = QHBoxLayout()**

**self.layv\_h3 = QHBoxLayout() # 歌单列表 #最低部，播放功能**

**self.layv\_h2\_v2 = QVBoxLayout() # buttomright 歌曲列表**

**self.layv\_h2\_v2\_h1 = QHBoxLayout() # 本地列表 扫描**

**self.layv\_h2\_v2\_h2 = QHBoxLayout() # 复选磁盘**

**self.layv\_h2\_v2\_h3 = QHBoxLayout() # 音乐列表和一个额外的列表**

**self.layv\_h2\_v2\_h3\_v1 = QVBoxLayout() # 额外的列表**

**self.layv\_h2\_v2\_h3\_v2 = QVBoxLayout() # 音乐列表**

**self.box1 = QPushButton('本地音乐')**

**self.box1.setCheckable(False)**

**self.box1.clicked.connect(self.box1f)**

**self.box2 = QPushButton('--->>播放列表<<---')**

**self.box2.setCheckable(True)**

**self.box2.clicked.connect(self.box2f)**

**self.layv\_h1.addWidget(self.box1)**

**# layv\_h1.addStretch(0)**

**self.layv\_h1.addWidget(self.box2)**

**# layv\_h1.addStretch(0)**

**self.layv.addLayout(self.layv\_h1)**

**self.layv.addStretch(0)**

**self.layv.addWidget(QLabel('\_' \* 500))**

**self.all, self.playlist, self.plist = {}, [], {}**

**self.readfile()**

**self.al = All(self.all)**

**self.label1 = QLabel('歌单', self)**

**self.addpl = QPushButton('新建')**

**self.addpl.setCheckable(False)**

**self.addpl.clicked.connect(self.addplf)**

**self.layv\_h2\_v1\_h1.addWidget(self.label1)**

**self.layv\_h2\_v1\_h1.addWidget(self.addpl)**

**self.pl = QListWidget()**

**self.pl.setMinimumSize(100, 350)**

**self.font = QFont()**

**self.font.setFamily('Arial')**

**self.font.setPointSize(18)**

**self.pl.setFont(self.font)**

**self.pl.addItem('like')**

**self.pl.addItems(self.playlist)**

**self.pl.setContextMenuPolicy(Qt.CustomContextMenu)**

**self.pl.customContextMenuRequested.connect(self.plr)**

**self.pl.doubleClicked.connect(self.dplf)**

**self.layv\_h2\_v1\_h3.addWidget(self.pl)**

**self.mlabel = QLabel('播放列表', self)**

**self.mlabel.setAlignment(Qt.AlignCenter)**

**self.layv\_h2\_v2.addWidget(self.mlabel)**

**self.layv\_h2\_v2.addStretch(0)**

**self.layv\_h2\_v2.addLayout(self.layv\_h2\_v2\_h1)**

**self.layv\_h2\_v2.addLayout(self.layv\_h2\_v2\_h2)**

**self.layv\_h2\_v2\_h3.addLayout(self.layv\_h2\_v2\_h3\_v1)**

**self.layv\_h2\_v2\_h3.addStretch(0)**

**self.layv\_h2\_v2\_h3.addLayout(self.layv\_h2\_v2\_h3\_v2)**

**self.layv\_h2\_v2\_h3.addStretch(10)**

**self.layv\_h2\_v2.addLayout(self.layv\_h2\_v2\_h3)**

**self.layv\_h2\_v2.addStretch(100)**

**self.ml = QTableWidget()**

**self.ml.setColumnCount(5)**

**self.ml.setRowCount(len(self.all['musiclist']))**

**self.ml.setEditTriggers(QAbstractItemView.NoEditTriggers)**

**self.ml.resizeColumnsToContents()**

**self.ml.resizeRowsToContents()**

**self.ml.setSelectionBehavior(QAbstractItemView.SelectRows)**

**self.ml.setShowGrid(False)**

**self.ml.setMinimumSize(500, 400)**

**self.ml.setColumnWidth(0, 50)**

**self.ml.setColumnWidth(1, 200)**

**self.ml.setColumnWidth(2, 200)**

**self.ml.setColumnWidth(3, 200)**

**self.ml.setColumnWidth(4, 70)**

**self.ml.setSelectionMode(QAbstractItemView.SingleSelection)**

**self.ml.setContextMenuPolicy(Qt.CustomContextMenu)**

**self.ml.customContextMenuRequested.connect(self.mlmenu)**

**self.ml.doubleClicked.connect(self.dmlf)**

**self.musicitems1, self.musicitems2, self.musicitems3 = [], [], []**

**self.musiclist(self.all['musiclist'])**

**self.layv\_h2\_v2\_h3\_v2.addWidget(self.ml)**

**self.layv\_h2\_v2\_h3\_v2.addStretch(10)**

**self.ml.verticalHeader().setVisible(False)**

**self.boxl = QPushButton('上一曲')**

**self.boxc = QPushButton('播放')**

**self.boxr = QPushButton('下一曲')**

**self.timelc = QLabel('0:00')**

**self.sec = self.all['playingtime']**

**self.control = QSlider(Qt.Horizontal) # 播放进程**

**self.musicname = QLabel(pa.Path(self.all['playingmusic']).stem)**

**self.musicmode = QLabel(self.all['playmode'])**

**self.boxvol = QSlider(Qt.Horizontal)**

**self.addtolike = QPushButton('☆')**

**if self.all['musicsec'] % 60 < 10:**

**sec = '0' + str(self.all['musicsec'] % 60)**

**else:**

**sec = str(self.all['musicsec'] % 60)**

**self.musicsec = QLabel(str(self.all['musicsec'] // 60) + ':' + sec)**

**self.layv\_h3\_h1 = QHBoxLayout()**

**self.layv\_h3\_h1.addWidget(self.musicname)**

**self.layv\_h3\_h1.addWidget(self.addtolike)**

**self.layv\_h3\_h1.addWidget(self.musicmode)**

**self.layv\_h3\_h1.addWidget(QLabel('音量'))**

**self.layv\_h3\_h1.addWidget(self.boxvol)**

**self.layv\_h3\_h1.addStretch(1)**

**self.layv\_h3.addWidget(self.boxl)**

**self.layv\_h3.addWidget(self.boxc)**

**self.layv\_h3.addWidget(self.boxr)**

**self.layv\_h3.addWidget(self.timelc)**

**self.layv\_h3.addWidget(self.control)**

**self.layv\_h3.addWidget(self.musicsec)**

**self.boxl.clicked.connect(self.boxlf)**

**self.boxc.clicked.connect(self.boxcf)**

**self.boxr.clicked.connect(self.boxrf)**

**self.control.setMaximum(self.all['musicsec'])**

**self.control.setValue(self.all['playingtime'])**

**self.control.sliderReleased.connect(self.controlf)**

**self.boxvol.setMaximum(100)**

**self.boxvol.setMaximumWidth(100)**

**self.boxvol.setValue(int(self.al.volume \* 100))**

**self.boxvol.sliderReleased.connect(self.volchange)**

**self.addtolike.clicked.connect(self.addtolikef)**

**if self.all['playingmusic'] in self.all['like']:**

**self.addtolike.setText('★')**

**com = QComboBox()**

**fontpath = pa.Path('C:\Windows\Fonts')**

**self.fonts = []**

**for x in fontpath.glob('\*\*/\*.ttf'):**

**self.fonts.append(x.stem)**

**com.addItems(self.fonts)**

**com.currentIndexChanged.connect(self.comf)**

**self.layv\_h2\_v1.addWidget(com)**

**com2 = QComboBox()**

**for x in range(10, 35):**

**com2.addItem(str(x))**

**com2.currentIndexChanged.connect(self.com2f)**

**com2.setCurrentText('15')**

**self.layv\_h2\_v1.addWidget(com2)**

**self.layv\_h2\_v1\_h3.addStretch(1)**

**self.layv\_h2\_v1.addLayout(self.layv\_h2\_v1\_h1)**

**self.layv\_h2\_v1.addStretch(0)**

**self.layv\_h2\_v1.addLayout(self.layv\_h2\_v1\_h2)**

**self.layv\_h2\_v1.addLayout(self.layv\_h2\_v1\_h3)**

**self.layv\_h2\_v1.addStretch(0)**

**self.layv\_h2.addLayout(self.layv\_h2\_v1)**

**self.layv\_h2.addStretch(0)**

**self.layv\_h2.addLayout(self.layv\_h2\_v2)**

**self.layv\_h2.addStretch(10)**

**self.layv.addLayout(self.layv\_h2)**

**self.layv.addStretch(1)**

**self.layv.addLayout(self.layv\_h3\_h1)**

**self.layv.addStretch(0)**

**self.layv.addLayout(self.layv\_h3)**

**self.layv.addStretch(0)**

**self.setLayout(self.layv)**

**self.show()**

**self.timer1 = QTimer()**

**self.timer1.timeout.connect(self.test)**

**self.timer = QTimer(self)**

**self.timer.timeout.connect(self.timerf)**

**self.timesec = QTimer()**

**self.timesec.timeout.connect(self.timesecf)**

**# 设置歌单字体**

**def comf(self, i):**

**print(self.fonts[i])**

**self.font.setFamily(self.fonts[i])**

**self.pl.setFont(self.font)**

**# 设置歌单字体大小**

**def com2f(self, i):**

**self.font.setPointSize(int(i) + 10)**

**self.pl.setFont(self.font)**

**# 打印信息**

**def clicked(self, qModelIndex):**

**QMessageBox.information(self, "ListWideget", "You chooce:" + self.qlist[qModelIndex.row()])**

**def clickeds(self, item):**

**QMessageBox.information(self, "ListWideget", "You chooce:" + item.text())**

**# 添加本地音乐**

**def box1f(self):**

**if not self.box1.isCheckable():**

**self.box1.setText('--->>本地音乐<<---')**

**self.box1.setCheckable(True)**

**self.box2.setText('播放列表')**

**self.box2.setCheckable(False)**

**self.box1\_1 = QPushButton('扫描磁盘音乐')**

**self.box1\_1.clicked.connect(self.box1\_1f)**

**self.box1\_1.setCheckable(False)**

**self.box1\_1.setToolTip('扫描文件')**

**self.box1\_2 = QPushButton('添加文件路径')**

**self.box1\_2.clicked.connect(self.box1\_2f)**

**self.box1\_2.setToolTip('添加文件路径')**

**self.layv\_h2\_v2\_h1.addWidget(self.box1\_1)**

**self.layv\_h2\_v2\_h1.addWidget(self.box1\_2)**

**self.mlabel.setText('本地音乐列表')**

**self.musiclist(self.all['allmusic'])**

**def musiclist(self, list):**

**self.ml.clear()**

**self.ml.setHorizontalHeaderLabels(['like', '歌曲', '歌手', '专辑', '时长'])**

**self.ml.setRowCount(len(list))**

**for y, x in enumerate(list):**

**try:**

**if x in self.all['like']:**

**new0 = QTableWidgetItem("★")**

**else:**

**new0 = QTableWidgetItem("☆")**

**if "-" in pa.Path(x).stem:**

**name = pa.Path(x).stem[pa.Path(x).stem.find('-') + 1:]**

**else:**

**name = pa.Path(x).stem**

**new1 = QTableWidgetItem(name.strip())**

**info = tinytag.TinyTag.get(x)**

**new2 = QTableWidgetItem(info.artist)**

**new3 = QTableWidgetItem(info.album)**

**if int(info.duration) % 60 < 10:**

**secs = '0' + str(int(info.duration) % 60)**

**else:**

**secs = str(int(info.duration) % 60)**

**times = str(int(info.duration // 60)) + ": " + secs**

**new4 = QTableWidgetItem(times)**

**self.ml.setItem(y, 0, new0)**

**self.ml.setItem(y, 1, new1)**

**self.ml.setItem(y, 2, new2)**

**self.ml.setItem(y, 3, new3)**

**self.ml.setItem(y, 4, new4)**

**except:**

**list.remove(x)**

**# 播放列表**

**def box2f(self):**

**if not self.box2.isCheckable():**

**if self.box1.isCheckable():**

**self.layv\_h2\_v2\_h1.removeWidget(self.box1\_1)**

**self.layv\_h2\_v2\_h1.removeWidget(self.box1\_2)**

**sip.delete(self.box1\_1)**

**sip.delete(self.box1\_2)**

**self.box2.setCheckable(True)**

**self.box1.setText('本地音乐')**

**self.box1.setCheckable(False)**

**self.box2.setText('--->>播放列表<<---')**

**self.mlabel.setText('播放列表')**

**self.musiclist(self.all['musiclist'])**

**# 存档**

**def readfile(self):**

**self.savefilename = 'save\_information.json'**

**if os.path.exists(self.savefilename):**

**print('exit')**

**with open(self.savefilename, 'r', encoding='gbk') as f:**

**self.all = json.load(f)**

**self.plist = self.all['playlist']**

**if not len(self.plist) == 0:**

**print(self.plist)**

**self.playlist = [x for x in self.plist.keys()]**

**else:**

**with open(self.savefilename, 'w', encoding='gbk') as f:**

**self.all = {'like': [], 'allmusic': [], 'playlist': {}, 'musiclist': [], 'playingmusic': '',**

**'playmode': '随机播放', 'playingtime': 0, 'playhistory': {}, 'musicsec': 0, 'volume': 1.0}**

**# 喜欢音乐列表，歌单，本地音乐列表，播放列表，播放模式，正在播放的音乐，已经播放的时间**

**# {'like':[],'playlist':{},'allmusic':[],'musiclist':[],'playmode':'随机播放','playingmusic':'','playingtime':0,'musicsec':'0.0','playhistory':{}}**

**json.dump(self.all, f)**

**# 新建歌单函数**

**def addplf(self):**

**if self.addpl.isCheckable():**

**print('T')**

**else:**

**print('F')**

**self.addpl.setCheckable(True)**

**self.plinput = QLineEdit()**

**self.plinput.setPlaceholderText('歌单名称')**

**self.okinput = QPushButton('OK')**

**self.okinput.clicked.connect(self.okinputf)**

**self.layv\_h2\_v1\_h2.addWidget(self.plinput)**

**self.layv\_h2\_v1\_h2.addWidget(self.okinput)**

**def okinputf(self):**

**print(self.plinput.text())**

**if not self.plinput.text() == '' and self.plinput.text() not in self.playlist and self.plinput.text() != 'like':**

**self.plist[self.plinput.text()] = []**

**self.playlist.append(self.plinput.text())**

**self.all['playlist'][self.plinput.text()] = []**

**self.pl.addItem(self.plinput.text())**

**elif self.plinput.text() in self.playlist:**

**print('歌单已存在')**

**self.layv\_h2\_v1\_h2.removeWidget(self.plinput)**

**self.layv\_h2\_v1\_h2.removeWidget(self.okinput)**

**sip.delete(self.plinput)**

**sip.delete(self.okinput)**

**self.addpl.setCheckable(False)**

**# 歌单右键事件**

**def plr(self):**

**cmenu = QMenu(self)**

**newAct = cmenu.addAction("新建")**

**delact = cmenu.addAction("删除")**

**action = cmenu.exec\_(QCursor.pos())**

**if action == newAct:**

**self.addplf()**

**elif action == delact and self.pl.currentRow() > 0:**

**self.plist.pop(self.playlist[self.pl.currentRow() - 1])**

**self.playlist.pop(self.pl.currentRow() - 1)**

**self.pl.takeItem(self.pl.currentRow())**

**# 本地音乐导入1**

**def box1\_1f(self):**

**if not self.box1\_1.isCheckable():**

**self.box2.setEnabled(False)**

**self.box1\_1.setCheckable(True)**

**self.box1\_2.setCheckable(True)**

**d = psutil.disk\_partitions()**

**self.gb = QGroupBox('选择你想要扫描的磁盘')**

**self.checkbox = QCheckBox()**

**list = []**

**for xx in d:**

**if not xx[3] == 'cdrom' and not xx[0][0] == 'C':**

**print(xx[0])**

**list.append(xx)**

**self.qc = [QCheckBox(xx[0][0], self) for xx in list]**

**for x in self.qc:**

**x.setChecked(False)**

**self.layv\_h2\_v2\_h2.addWidget(x)**

**self.okscan = QPushButton('OK')**

**self.okscan.clicked.connect(self.okscanf)**

**self.layv\_h2\_v2\_h2.addWidget(self.okscan)**

**self.gb.setLayout(self.layv\_h2\_v2\_h2)**

**self.layv\_h2\_v2\_h2.insertWidget(0, self.gb)**

**# 本地音乐导入2**

**def box1\_2f(self):**

**if not self.box1\_2.isCheckable():**

**self.box2.setEnabled(False)**

**self.box1\_2.setCheckable(True)**

**self.box1\_1.setCheckable(True)**

**filed = QFileDialog()**

**filed.setFileMode(QFileDialog.Directory)**

**name = str(filed.getExistingDirectory())**

**print(name)**

**self.scanpath(name)**

**def okscanf(self):**

**self.timer.start(100)**

**self.okscan.setText('wait...')**

**self.okscan.setEnabled(False)**

**self.box1\_2.setEnabled(False)**

**def timerf(self):**

**self.timer.stop()**

**self.okscan.setEnabled(True)**

**for x in self.qc:**

**if x.isChecked():**

**path = x.text() + ':\\'**

**print(path)**

**self.scanpath(path)**

**self.gb.setTitle('扫描完毕')**

**self.layv\_h2\_v2\_h2.removeWidget(self.gb)**

**sip.delete(self.gb)**

**for x in self.qc:**

**self.layv\_h2\_v2\_h2.removeWidget(x)**

**sip.delete(x)**

**self.layv\_h2\_v2\_h2.removeWidget(self.okscan)**

**self.okscan.setText('OK')**

**sip.delete(self.okscan)**

**self.box2.setEnabled(True)**

**self.box1\_2.setEnabled(True)**

**self.box1\_1.setCheckable(False)**

**self.box1\_2.setCheckable(False)**

**# 扫描音频文件**

**def scanpath(self, path):**

**for xx in pa.Path(path).glob('\*\*/\*'):**

**if tinytag.TinyTag.is\_supported(str(xx.name)):**

**if (os.path.getsize(xx)) > 1 \* 1024 \* 1024 and (**

**os.path.getsize(xx)) < 50 \* 1024 \* 1024 and str(xx) not in self.all['allmusic']:**

**print(xx)**

**self.all['allmusic'].append(str(xx))**

**self.musiclist(self.all['allmusic'])**

**self.box2.setEnabled(True)**

**self.box1\_2.setEnabled(True)**

**self.box1\_1.setCheckable(False)**

**self.box1\_2.setCheckable(False)**

**# 音乐列表中右键事件**

**def mlmenu(self, pos):**

**print(self.ml.currentRow())**

**if self.ml.currentRow() >= 0:**

**menu = QMenu()**

**item1 = menu.addAction("播放")**

**item2 = menu.addAction("暂停")**

**item3 = menu.addAction('下一曲')**

**menu1 = QMenu()**

**menu1.setTitle('播放模式')**

**m, modes = ['顺序播放', '列表循环', '单曲循环', '随机播放'], []**

**for x in m:**

**if self.all['playmode'] == x:**

**modes.append(menu1.addAction('✔' + x))**

**else:**

**modes.append(menu1.addAction(' ' + x))**

**menu.addMenu(menu1)**

**item4 = menu.addAction('标记为喜欢')**

**menu2 = QMenu()**

**menu2.setTitle('添加到歌单')**

**menu.addMenu(menu2)**

**menu3 = QMenu()**

**menu34 = QMenu()**

**menu35 = QMenu()**

**menu3.addMenu(menu34)**

**menu3.addMenu(menu35)**

**menu.addMenu(menu3)**

**menu3.setTitle('添加同')**

**menu34.setTitle('歌手到')**

**menu35.setTitle('专辑到')**

**menu4 = QMenu()**

**menu.addMenu(menu4)**

**menu4.setTitle('删除同')**

**item41 = menu4.addAction('歌手')**

**item42 = menu4.addAction('专辑')**

**pls, pls34, pls35 = [], [menu34.addAction('like')], [menu35.addAction('like')]**

**for x in self.all['playlist'].keys():**

**pls.append(menu2.addAction(x))**

**pls34.append(menu34.addAction(x))**

**pls35.append(menu35.addAction(x))**

**item5 = menu.addAction('删除')**

**item6 = menu.addAction('在文件管理器中打开')**

**action = menu.exec\_(self.ml.mapToGlobal(pos))**

**if action == item1: # 播放**

**self.dmlf()**

**elif action == item2: # 暂停**

**mixer.music.pause()**

**self.timer1.stop()**

**self.boxc.setText('暂停')**

**self.timesec.stop()**

**elif action == item3: # 下一曲**

**self.test()**

**elif action in modes:**

**self.all['playmode'] = m[modes.index(action)]**

**self.musicmode.setText(self.all['playmode'])**

**elif self.box2.isCheckable(): # 播放列表下**

**self.operatorlist(action, self.all['musiclist'], pls, pls34, pls35, item4, item41, item42,**

**item5, item6)**

**elif self.box1.isCheckable():**

**self.operatorlist(action, self.all['allmusic'], pls, pls34, pls35, item4, item41, item42,**

**item5, item6)**

**elif self.mlabel.text() in self.all['playlist'].keys():**

**self.operatorlist(action, self.all['playlist'][self.mlabel.text()], pls, pls34, pls35, item4,**

**item41, item42, item5, item6)**

**elif self.mlabel.text() == 'like':**

**self.operatorlist(action, self.all['like'], pls, pls34, pls35, item4, item41, item42,**

**item5, item6)**

**def operatorlist(self, action, list, pls, pls34, pls35, item4, item41, item42, item5, item6):**

**ti = tinytag.TinyTag.get(list[self.ml.currentRow()]) # 音乐的所有信息全包括在内了**

**parents = pa.Path(list[self.ml.currentRow()]).parent**

**if action in pls: # 添加到指定歌单**

**if list[self.ml.currentRow()] not in self.all['playlist'][self.playlist[pls.index(action)]]:**

**self.all['playlist'][self.playlist[pls.index(action)]].append(**

**list[self.ml.currentRow()])**

**elif action in pls34:**

**for x in list:**

**tx = tinytag.TinyTag.get(x)**

**if tx.artist == ti.artist and pls34.index(action) == 0 and x not in self.all['like']:**

**self.all['like'].insert(0, x)**

**elif tx.artist == ti.artist and pls34.index(action) != 0 and x not in self.all['playlist'][**

**self.playlist[pls34.index(action) - 1]]:**

**self.all['playlist'][self.playlist[pls34.index(action) - 1]].insert(0, x)**

**elif action in pls35:**

**for x in list:**

**tx = tinytag.TinyTag.get(x)**

**if tx.album == ti.album and pls35.index(action) == 0 and x not in self.all['like']:**

**self.all['like'].insert(0, x)**

**elif tx.album == ti.album and pls34.index(action) != 0 and x not in self.all['playlist'][**

**self.playlist[pls35.index(action) - 1]]:**

**self.all['playlist'][self.playlist[pls35.index(action) - 1]].insert(0, x)**

**elif action == item4: # add to like**

**if list[self.ml.currentRow()] not in self.all['like']:**

**self.all['like'].append(list[self.ml.currentRow()])**

**elif action == item41:**

**for x in list:**

**tx = tinytag.TinyTag.get(x)**

**if tx.artist == ti.artist:**

**list.remove(x)**

**elif action == item42:**

**for x in list:**

**tx = tinytag.TinyTag.get(x)**

**if tx.album == ti.album:**

**list.remove(x)**

**elif action == item5:**

**list.pop(self.ml.currentRow())**

**elif action == item6:**

**file = QFileDialog()**

**name = file.getOpenFileName(self, 'open file', str(parents), "Music file(\*.mp3 \*.flac)")**

**self.musiclist(list)**

**# 点击☆/★收藏或取消收藏**

**def addtolikef(self):**

**if self.all['playingmusic'] not in self.all['like']:**

**self.all['like'].insert(0, self.all['playingmusic'])**

**self.addtolike.setText('★')**

**elif self.all['playingmusic'] in self.all['like']:**

**self.all['like'].remove(self.all['playingmusic'])**

**self.addtolike.setText('☆')**

**if self.mlabel.text() == 'like':**

**self.musiclist(self.all['like'])**

**# 播放暂停**

**def playmusic(self, filename, start=0): # 播放**

**ting = mixer.music.load(filename)**

**info = tinytag.TinyTag.get(filename)**

**mixer.music.play(start=start)**

**self.musicname.setText(str(pa.Path(filename).stem))**

**if int(info.duration) % 60 < 10:**

**secs = '0' + str(int(info.duration) % 60)**

**else:**

**secs = str(int(info.duration) % 60)**

**self.musicsec.setText(str(int(info.duration) // 60) + ':' + secs)**

**self.timer1.start(int((info.duration - start) \* 1000))**

**self.ml.selectRow(self.all['musiclist'].index(filename))**

**self.boxc.setText('正在播放')**

**self.all['playingmusic'] = filename**

**self.sec = start**

**self.all['musicsec'] = int(info.duration)**

**self.timesec.start(1000)**

**self.control.setMaximum(int(info.duration))**

**self.control.setValue(start)**

**# 下一曲**

**def test(self):**

**if self.all['playmode'] == '顺序播放':**

**if self.all['playingmusic'] != self.all['musiclist'][-1]:**

**file = self.all['musiclist'][self.all['musiclist'].index(self.all['playingmusic']) + 1]**

**elif self.all['playmode'] == '列表循环':**

**if len(self.all['musiclist'])==self.all['musiclist'].index(self.all['playingmusic'])+1:**

**file = self.all['musiclist'][0]**

**else:**

**file = self.all['musiclist'][self.all['musiclist'].index(self.all['playingmusic']) + 1]**

**elif self.all['playmode'] == '单曲循环':**

**file = self.all['playingmusic']**

**else:**

**file = self.all['musiclist'][numpy.random.randint(len(self.all['musiclist']))]**

**self.playmusic(file)**

**# 上一曲**

**def last(self):**

**if self.all['playmode'] == '随机播放':**

**file = self.all['musiclist'][numpy.random.randint(len(self.all['musiclist']))]**

**else:**

**file = self.all['musiclist'][self.all['musiclist'].index(self.all['playingmusic']) - 1]**

**self.playmusic(file)**

**# 双击播放列表**

**def dmlf(self):**

**if self.box1.isCheckable() and not self.box2.isCheckable():**

**self.all['musiclist'] = self.all['allmusic']**

**elif not self.box1.isCheckable() and not self.box2.isCheckable():**

**if self.mlabel.text() == 'like':**

**self.all['musiclist'] = self.all['like']**

**else:**

**self.all['musiclist'] = self.all['playlist'][self.mlabel.text()]**

**self.playmusic(self.all['musiclist'][self.ml.currentRow()])**

**# 双击打开歌单**

**def dplf(self):**

**if self.box2.isEnabled():**

**if self.box1.isCheckable():**

**self.layv\_h2\_v2\_h1.removeWidget(self.box1\_1)**

**self.layv\_h2\_v2\_h1.removeWidget(self.box1\_2)**

**sip.delete(self.box1\_1)**

**sip.delete(self.box1\_2)**

**self.box1.setText('本地列表')**

**self.box2.setText('播放列表')**

**self.box1.setCheckable(False)**

**self.box2.setCheckable(False)**

**if self.pl.currentRow() != 0:**

**self.mlabel.setText(self.playlist[self.pl.currentRow() - 1])**

**self.musiclist(self.all['playlist'][self.playlist[self.pl.currentRow() - 1]])**

**else:**

**self.mlabel.setText('like')**

**self.musiclist(self.all['like'])**

**def timesecf(self):**

**self.sec += 1**

**if self.sec % 60 < 10:**

**secs = '0' + str(self.sec % 60)**

**else:**

**secs = str(self.sec % 60)**

**self.timelc.setText(str(self.sec // 60) + ':' + secs)**

**self.control.setValue(self.sec)**

**if self.all['playingmusic'] in self.all['like']:**

**self.addtolike.setText('★')**

**else:**

**self.addtolike.setText('☆')**

**if self.sec == int(self.all['musicsec'] / 2):**

**if self.all['playingmusic'] in self.all['playhistory']:**

**self.all['playhistory'][self.all['playingmusic']] += 1**

**else:**

**self.all['playhistory'][self.all['playingmusic']] = 1**

**def boxlf(self):**

**if self.all['playingmusic'] != '':**

**self.last()**

**def boxcf(self):**

**if self.boxc.text() == '播放':**

**if self.all['playingmusic'] != '':**

**self.playmusic(self.all['playingmusic'], start=self.all['playingtime'])**

**elif len(self.all['musiclist']) != 0:**

**self.playmusic(self.musiclist[0])**

**elif self.boxc.text() == '暂停':**

**# self.playmusic(self.all['playingmusic'],start=self.sec)**

**mixer.music.unpause()**

**self.boxc.setText('正在播放')**

**self.timesec.start(1000)**

**self.timer1.start((self.all['musicsec'] - self.sec) \* 1000)**

**elif self.boxc.text() == '正在播放':**

**self.timer1.stop()**

**self.timesec.stop()**

**mixer.music.pause()**

**self.boxc.setText('暂停')**

**def boxrf(self):**

**if self.all['playingmusic'] != '':**

**self.test()**

**# 进度条控制**

**def controlf(self):**

**self.playmusic(self.all['playingmusic'], start=self.control.value())**

**# 声音大小调整**

**def volchange(self):**

**mixer.music.set\_volume(float(self.boxvol.value()) / 100)**

**class MainWindow(QMainWindow):**

**def \_\_init\_\_(self):**

**super().\_\_init\_\_()**

**self.initt()**

**def initt(self):**

**exitAction = QAction('&退出', self)**

**exitAction.setShortcut('关闭程序')**

**exitAction.setStatusTip('Exit application')**

**exitAction.triggered.connect(self.save\_quit)**

**self.statusBar()**

**menubar = self.menuBar()**

**fileMenu = menubar.addMenu('&文件')**

**fileMenu.addAction(exitAction)**

**self.toolbar = self.addToolBar('')**

**la = QLabel()**

**self.resize(1000, 610) # 500 400**

**screen = QDesktopWidget().screenGeometry()**

**size = self.geometry()**

**self.move((screen.width() - size.width()) / 2, (screen.height() - size.height()) / 2)**

**self.lay = ttt(self)**

**self.lay.move(0, 50)**

**self.lay.resize(size.width(), size.height())**

**self.setWindowTitle('音乐播放器')**

**self.show()**

**def closeEvent(self, event):**

**self.save()**

**event.accept()**

**# 保存**

**def save(self):**

**print('exit')**

**self.lay.all['playlist'] = self.lay.plist**

**self.lay.all['playingtime'] = self.lay.sec**

**self.lay.all['volume'] = mixer.music.get\_volume()**

**print(self.lay.all)**

**print(self.lay.savefilename)**

**with open(self.lay.savefilename, 'w', encoding='gbk') as f:**

**json.dump(self.lay.all, f)**

**# self.all = {'like': [], 'playlist': {}, 'musiclist': [], 'playmode': 'mode 4', 'playingmusic': '','playingtime': 0}**

**# like[] playlist{} playmode str musiclist [] playingmusic str playingtime int ,playhistory {} playingsec?**

**# 喜欢音乐列表，歌单，播放列表，播放模式，正在播放的音乐，已经播放的时间**

**# 退出**

**def save\_quit(self):**

**self.save()**

**qApp.quit()**

**def resizeEvent(self, evt):**

**size = self.geometry()**

**self.lay.resize(size.width(), size.height() - 50)**

**self.lay.ml.setMinimumSize(size.width(), size.height() - 220)**

**self.lay.pl.setMinimumHeight(size.height() - 290)**

**if \_\_name\_\_ == "\_\_main\_\_":**

**app = QApplication(sys.argv)**

**window = MainWindow()**

**sys.exit(app.exec\_())**

2、json文件

该文件主要实现了存储信息功能，具体代码如下：

{"like": ["D:\\CloudMusic\\Jam - \u4e0d\u9732\u58f0\u8272.mp3", "D:\\CloudMusic\\Fool's Garden - Lemon Tree.mp3", "D:\\CloudMusic\\Ice Paper - \u5fc3\u5982\u6b62\u6c34.mp3", "D:\\CloudMusic\\Kirsty\u5218\u747e\u777f - \u82e5\u628a\u4f60.mp3", "D:\\CloudMusic\\\u4e00\u652f\u69b4\u83b2 - \u6d77\u5e95.mp3"],

"allmusic": ["D:\\CloudMusic\\Fool's Garden - Lemon Tree.mp3", "D:\\CloudMusic\\Ice Paper - \u5fc3\u5982\u6b62\u6c34.mp3", "D:\\CloudMusic\\Jam - \u4e0d\u9732\u58f0\u8272.mp3", "D:\\CloudMusic\\Kirsty\u5218\u747e\u777f - \u82e5\u628a\u4f60.mp3", "playlist": {"aa": ["D:\\CloudMusic\\Jam - \u4e0d\u9732\u58f0\u8272.mp3"]}, "musiclist": ["D:\\CloudMusic\\Fool's Garden - Lemon Tree.mp3", "D:\\CloudMusic\\Ice Paper - \u5fc3\u5982\u6b62\u6c34.mp3","D:\\CloudMusic\\\u4e09\u65e0MarBlue - \u8fbe\u62c9\u5d29\u5427\uff08Cover \u6d1b\u5929\u4f9d&\u8a00\u548c\uff09.mp3", "D:\\CloudMusic\\\u4e09\u66f4\u706f\u706b - \u80e1\u5e7f\u751f.mp3", "D:\\CloudMusic\\\u5341\u4e09Asa - \u5173\u952e\u8bcd(\u6296\u97f3\u70ed\u641c\u7248).mp3", "D:\\CloudMusic\\\u5468\u534e\u5065 - \u96be\u5ff5\u7684\u7ecf.mp3", "playingmusic": "D:\\CloudMusic\\Ice Paper - \u5fc3\u5982\u6b62\u6c34.mp3", "playmode": "\u5217\u8868\u5faa\u73af",

"playingtime": 74,

"playhistory":{"D:\\CloudMusic\\VH(Vast&Hazy)-\u4e0e\u6d6a\u4e4b\u95f4.mp3": 2, "D:\\CloudMusic\\Jam - \u4e0d\u9732\u58f0\u8272.mp3": 1, "D:\\CloudMusic\\Kirsty\u5218\u747e\u777f - \u82e5\u628a\u4f60.mp3": 1}, "musicsec": 185, "volume": 0.9921875}