# Pygame 音乐播放器(PMP)

想法来源：

一度缺乏想象力的我，不管如何绞尽脑汁，要不然就是想法太烂，要不然缺乏趣味。于是苦闷的我只好放弃做游戏的这一想法，打开了网易云，默默的戴上耳机，从心灵中感受灵感的魂发。

灵感来自于现实，这时正在听网易云的我，突然灵机一动，看了眼网易云，正好一直热爱听歌的我，播放器不就是一个实用的软件嘛。于是就这样我便由衷的搞出了个远大的想法——音乐播放器。

名字想法：

简单明了。

题外话

这个项目也总计写了我差不多64个小时(代码55小时, 文档6小时，画图 3小时左右) 拖拖拉拉 一个一个功能写也正好半个暑假(一个月)左右，算是我写过最长的时间吧。

虽然花费了这么多时间去写，但是不管是哪个项目，写完的那一刻回头一看说实话收获了很多。数次的调试、琢磨，从0KB到39KB，一天花尽个3-6小时绞尽脑汁一行行的刻印上去，感觉就像在建造一个生命体，构造他的灵魂，给我带来无穷的经验……

程序我是懒得（没那么多时间）维护了，这1000行的代码有很多可以优化的地方（当然bug是没有的 最多有一处 最后一页会提到），还有很多可以增加的地方，如果以后有空我会继续完善这个生命体的。

感谢家长与老师的支持，我会继续努力的，也感谢日日夜夜陪伴我的网易(抑)云233，希望这个项目能有一个出色的成绩(虽然我猜测比我强大的项目已不在少数)

2020-8-30 21:40

符

符

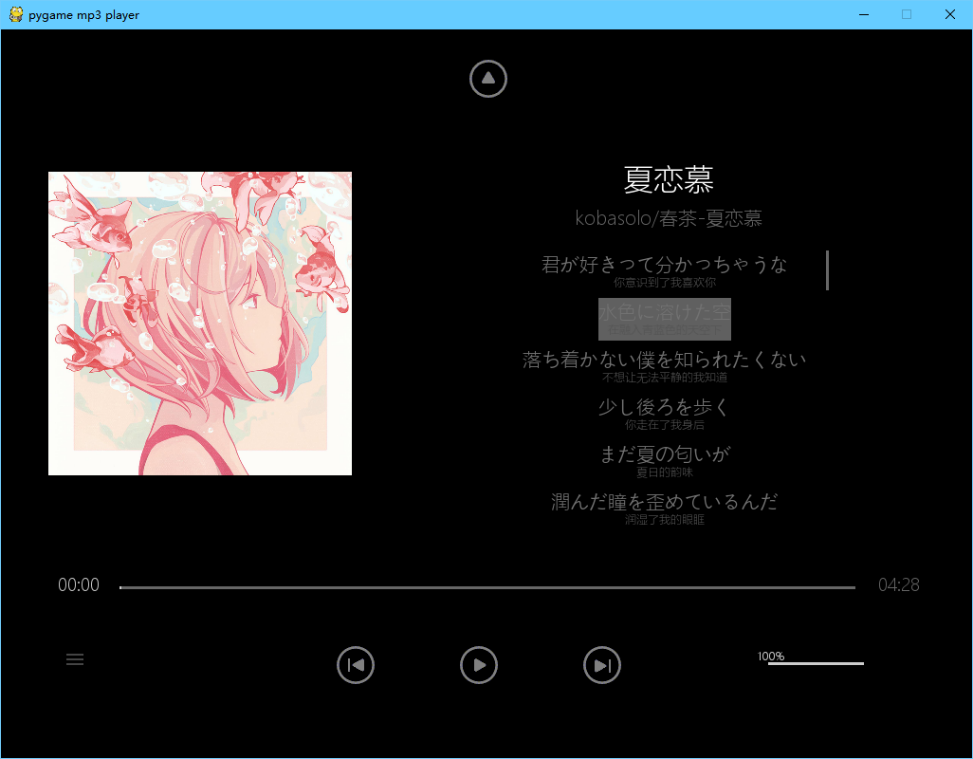
## UI：

窗口大小: 1024 \* 768

### 音乐列表：

****

### 播放界面：

****

## 各个文件及文件夹说明:

1. **font**: 存放字体文件夹
2. **img**: 存放图片资源文件夹
3. **lyric**: 存放歌词文件夹
4. **music**: 存放mp3音乐的文件夹
5. **temp**: 临时存放封面图片资源的文件夹
6. **NOBUG.txt**: 奇怪的字符画
7. **PygameMp3Player.py**: 主运行代码
8. **TestReadLryic.py**: 残留的测试代码

## 操作事例：

### 添加音乐:

1. 找到程序根目录下的“music”文件夹中放入mp3格式的音乐即可(仅支持mp3格式的音乐 其他格式将被无视)

### 添加歌词:

1. 找到程序根目录下的“lyric”文件夹

2. 命名格式: 对应mp3文件名(不需要后缀) + .lrc

3. 将文件放入当前文件夹

### 添加翻译歌词（必须有原歌词）:

1. 找到程序根目录下的“lyric”文件夹

2. 命名格式: 对应mp3文件名(不需要后缀) + \_tran + .lrc

3. 将文件放入当前文件夹

#### 例子:





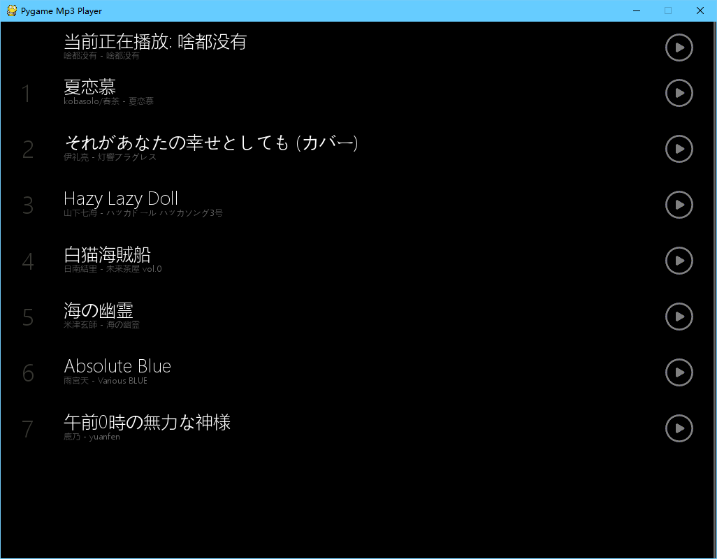
#### 启动程序:

1. 输入目标音乐文件夹(默认为程序根目录下的music文件夹)

 2. 输入目标歌词文件夹(默认为程序根目录下的lyric文件夹)

#### 音乐列表:

1. 当单页歌曲超过6首，可以使用滚轮进行翻页

 2. 点击播放界面播放对应音乐，点击“当前正在播放:”右边的播放键即跳转播放界面

#### 播放列表:

粉色：返回音乐列表

橙色: 可以单击目标歌词从而跳转到指定位置(根据lrc位置来定

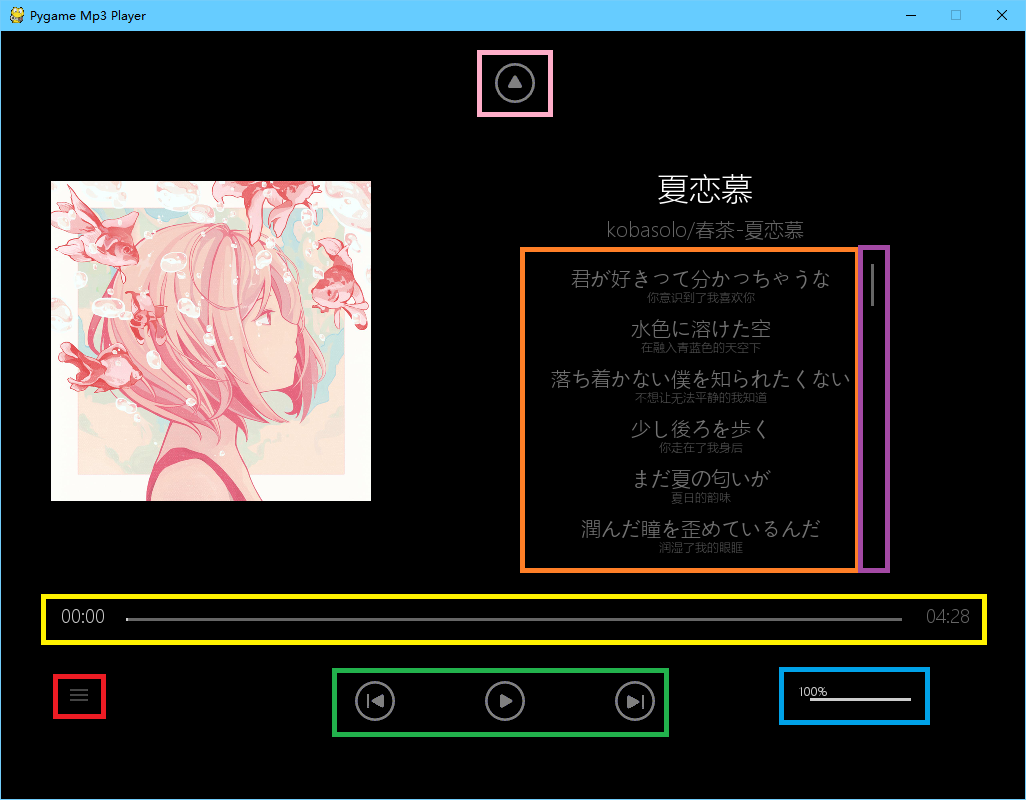
紫色: 滚轮翻页（按键盘按键’R’重置到正在播放的歌词位置）

黄色: 拖动进度条以便跳转到任意位置

红色: 单击以切换播放模式(顺序、循环、随机)

绿色: 歌曲控制台(我没来得及和懒地写快捷键了…)

蓝色: 拖动音量条以便更改软件输出音量



#### 资源文件及模块(库):

##### Img文件夹:

1.资源借鉴说明: 其中借鉴了网易云音乐的播放键其余资源为PS完成

2. (每个图片看一下就明白了，所以不作说明)

**外部库:**

eyed3: 读取mp3数据(如封面、歌曲名、歌手等)

pygame: …

#### 程序结构

**类:**

*class* PygameMp3Player(*object*) *# FrameWork*

*# 管理*

*class* PlayManager(*object*) *# 播放管理(如暂停、播放时间等)*

*class* MusicTimeManager(*object*) *# 播放时间管理*

*~~class~~* ~~ThemeManager(~~*~~object~~*~~)~~ *~~# 皮肤管理(未实现)~~*

*# 存储数据对象*

*~~class~~* ~~Theme(~~*~~object~~*~~)~~ *~~# 单个皮肤对象(未实现)~~*

*class* Music(*object*) *# 单个mp3对象*

*# 控件*

*class* ScrollBar(*object*) *# 滚动条*

*class* Button(*object*) *# 按钮*

*class* PlayBar(*object*) *# 播放时间进度条*

*class* VolumeBar(*object*) *# 音量条*

*class* MusicList(*object*) *# 单个列表对象*

*class* SpecialMusicList(MusicList) *# 特殊的列表对象(即“正在播放: ”)*

*class* LyricDisplay(*object*) *# 歌词显示空间*

*# 显示*

*class* PlayInfoDisplay(*object*) *# 播放界面的信息显示(封面 歌曲名等)*

*# UI*

*class* MusicListUI(*object*) *# 列表界面UI*

*class* PlayUI(*object*) *# 播放界面UI*

*# 歌词 & 容器*

*class* Lyric(*object*) *# 歌词文件中一行的数据*

*class* LyricDisplayObj(*object*) *# 歌词文件中一行的数据对象(用于显示)*

*class* LyricContainer(*object*) *# 存放Lyric对象容器*

*class* LyricDisplayObjContainer(*object*)*# 存放LyricDisplayObj对象容器*

**流程:**

**启动程序:**

**输入: 文件夹路径(音乐 & 歌词)**

1. PygameMp3Player初始化
2. 遍历当前的文件夹中的文件, 如果为mp3文件则实现Music对象并append至音乐列表(musicLst)中
3. 初始化两个UI并加入列表中(MusicListUI PlayUI)
4. 等待用户操作…

**切换到另一个UI界面:**

1. 更改PygameMp3Player对象中的status值

*44 self*.status = 0 *# 0: 音乐列表, 1: 播放界面*

2. 控制对应UI 比如

*68 self*.uiLst[*self*.status].show(*self*.scr)

**播放时间控制(底层由时间戳实现):**

*965 def* get\_current\_time\_millis():  
*966*  *return int*(*round*(time.time() \* 1000))

**切换到新的一首歌(重置为0秒):**

1. 更改播放起始时间戳为当前的进度

*213**self*.beginPlayMs = get\_current\_time\_millis()

**跳转到指定秒数:**

1. 新的进度(秒) = 当前已播放秒数 - **(**新的秒数 – 已经播放的秒数**)**

*194 self*.beginPlayMs -= (sec \* 1000 - *self*.ms\_played)

**同步秒数(暂停后进行播放操作)**

1.当前的时间戳 – 已经播放的进度

*187 self*.beginPlayMs = get\_current\_time\_millis() - *self*.ms\_played

**用到的数据结构:**

功能: 播放后退

数据结构: 栈 Stack

大致原理:

播放完一首歌曲： 推入(push)当前歌曲下标

*120 self*.previousLst.append(*self*.musicIndex)

播放上一首歌: 弹出(pop)并获取最后一首歌的歌曲下标

*132 self*.switch(*self*.previousLst.pop())

Bug的话就是如果程序失去焦点太久 再获取焦点的话 那个进度条可能会很长 不过换个歌就可以了（主要是播放时间是由时间戳实现的emmm）