**南京航空航天大学**

《面向对象程序设计语言》课程设计报告

**基于Python的音乐播放器**

**学号：161510123**

**姓名：张天弈**

**日期：2016-5-14**

**目 录**

[一、需求分析： 3](file:///G:\My%20Documents\Documents%20and%20Settingslhf桌面2002-2003学年第一学期课程设计（课程实习与大作业）课程实习及大作业（可挂网）C++%22%20l#_Toc27251728)

[二、程序的主要功能： 3](C:Documents%20and%20Settingslhf桌面2002-2003学年第一学期课程设计（课程实习与大作业）课程实习及大作业（可挂网）C++%22%20l%20)

[三、程序运行平台： 3](C:Documents%20and%20Settingslhf桌面2002-2003学年第一学期课程设计（课程实习与大作业）课程实习及大作业（可挂网）C++%22%20l%20)

[四、 系统总框架图 3](C:Documents%20and%20Settingslhf桌面2002-2003学年第一学期课程设计（课程实习与大作业）课程实习及大作业（可挂网）C++%22%20l%20)

[五、程序类的说明： 3](C:Documents%20and%20Settingslhf桌面2002-2003学年第一学期课程设计（课程实习与大作业）课程实习及大作业（可挂网）C++%22%20l%20)

[六、模块分析 5](C:Documents%20and%20Settingslhf桌面2002-2003学年第一学期课程设计（课程实习与大作业）课程实习及大作业（可挂网）C++%22%20l%20)

[七、比较有特色的函数 6](C:Documents%20and%20Settingslhf桌面2002-2003学年第一学期课程设计（课程实习与大作业）课程实习及大作业（可挂网）C++%22%20l%20)

[八、存在的不足与对策 7](C:Documents%20and%20Settingslhf桌面2002-2003学年第一学期课程设计（课程实习与大作业）课程实习及大作业（可挂网）C++%22%20l%20)

[九、程序源代码 7](C:Documents%20and%20Settingslhf桌面2002-2003学年第一学期课程设计（课程实习与大作业）课程实习及大作业（可挂网）C++%22%20l%20)

# 一、需求分析

需要一个纯净不推送信息的音乐播放器

Python具有许多外部图形处理库，比如Pygame

并且Pygame具有基于ffmpeg的音频处理接口，适合进行音乐播放

# 二、程序的主要功能

1、音乐功能：

播放source目录下的全部音乐

支持MP3, OGG, WAV格式的音频

可以用上下箭头或点击音量图标控制音量

可以用左右箭头或点击前后图标切换音乐

可以用空格键开始或暂停音乐

按停止键有2.5秒淡出效果

2、编辑功能：

可添加或删除音乐

需自行输入添加音乐的路径

3、用户功能：

有用户注册及登陆功能

由正则表达式匹配用户名与密码

可选择是否注册账户

用户信息存放在info/user . pkl文件中

点击右上角设置键可查看用户的名称和密码对应的MD5值

4、喜好功能：

可点击左下角红心代表喜好歌曲

5、图形界面：

背景设为一张唱片

6、多语言功能：

可以切换中英文

7、加密功能：

使用MD5方法进行用户数据加密

# 三、程序运行平台

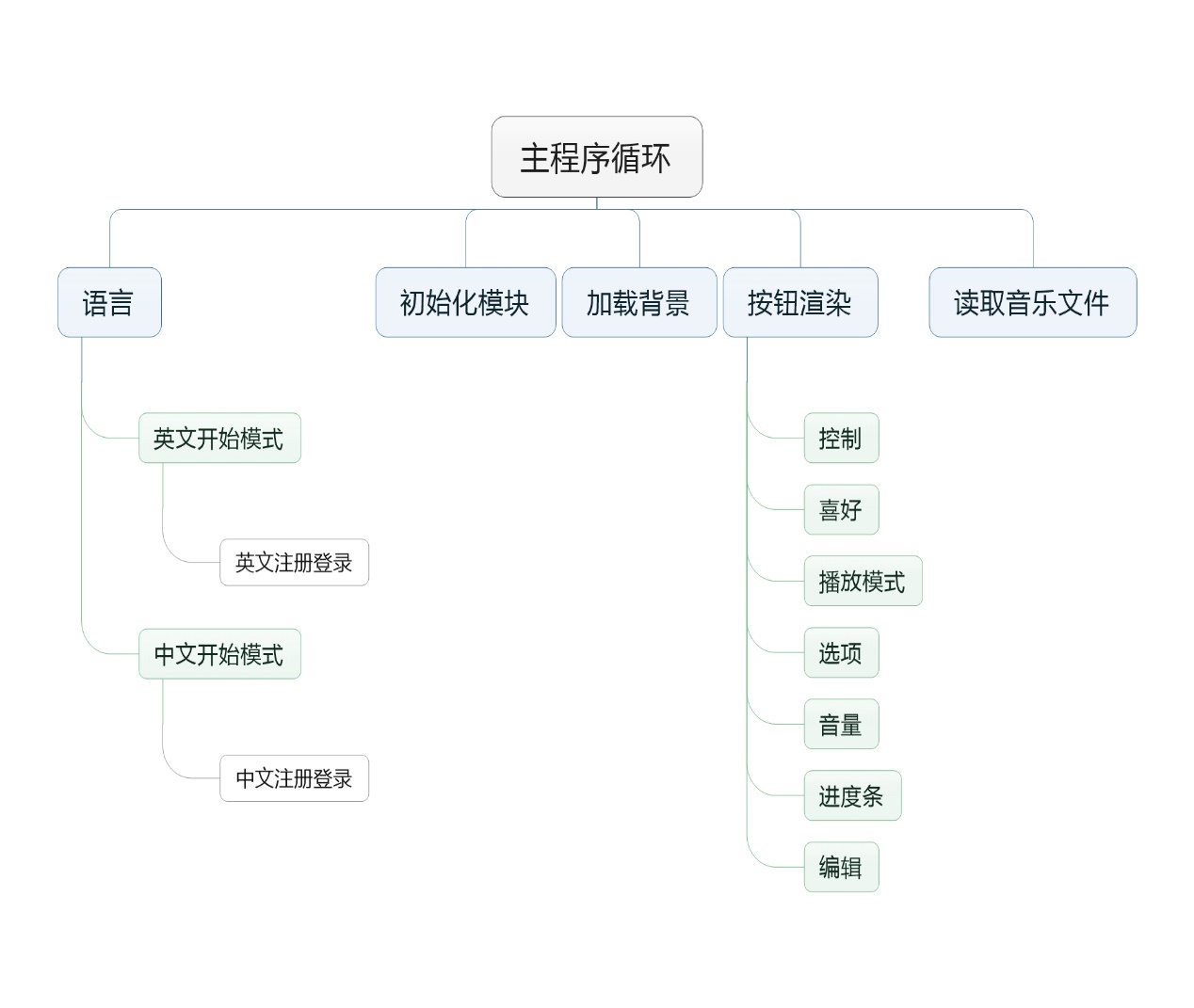
Windows系统可点击根目录下的exe文件运行

也可以在安装好Python3.4后选择IDLE进入调试模式

还可下载Pycharm进行调试

点击工具栏的运行模式运行根目录下的main.py

# 四、 系统总框架图



# 五、程序类的说明

# 语言  
**class Language(**object**):  
  
 def \_\_init\_\_(**self**):# 初始化函数  
 pass  
  
 @**staticmethod  
 **def choose(**self**):# 选择函数** num **=** input**('1 English, 2 中文:')  
 return** num # 返回输入值  
  
  
# 英文开始模式  
**class Select(**object**):  
  
 def \_\_init\_\_(**self**):# 初始化函数  
 pass  
  
 @**staticmethod  
 **def choose(**self**):# 选择函数** num **=** input**('1 to create an account, 2 to log in, 3 to start without logging in:')  
 return** num # 返回输入值  
  
  
# 中文开始模式  
**class SelectChinese(**object**):  
  
 def \_\_init\_\_(**self**): # 初始化函数  
 pass  
  
 @**staticmethod  
 **def choose(**self**): # 选择函数** num **=** input**('按1创建账户, 按2登录, 按3不登录直接启动:')  
 return** num # 返回输入值  
  
  
# 英文注册登录  
**class Registration(**object**):  
  
 def \_\_init\_\_(**self**): # 初始化函数** self.username **= ''# 初始化用户名** self.password **= ''# 初始化密码** # 注册  
 **def start(**self**):** username **=** input**(  
 'Please create the username(Must be 3-10 characters or numbers, no Chinese or space allowed):')** # 正则匹配  
 **if** re.match**(r'\w{3,10}'**, username**):** self.username **=** username  
 **else:** print**('Input error!')** # 输入错误则重新输入  
 **while** self.username **!=** username**:** self.start**()** password **=** input**(  
 'Please create the password(Must be 6-20 characters or numbers, no Chinese or space allowed):')** # 正则匹配  
 **if** re.match**(r'\w{6,20}'**, password**):** self.password **=** password  
 # 输入错误则重新输入  
 **while** self.password **!=** password**:** self.start**()** # 静态方法求MD5  
 **@**staticmethod  
 **def getmd5(**s**):** md5 **=** hashlib.md5**()** md5.update**(**s.encode**('utf-8'))  
 return** md5.hexdigest**()**

**# 注册  
 def register(**self**):** self.start**()** # MD5加密  
 db**[**self.username**] =** self.getmd5**(**self.password **+** self.username**)** # 将信息通过dict类型格式化二进制存入文件中  
 \_file **=** open**('info/user.pkl'**, **'wb')** data **= {'Username': (**self.getmd5**(**self.username**)**, **u'Unicode')**, **'Password': (**self.getmd5**(**self.password**)**, **u'Unicode')}  
 try:** pickle.dump**(**data, \_file**)  
 finally:** \_file.close**()** print**('Registration successful！')** # 登录  
 **def login(**self**):** username **=** input**('Please input the username:')** password **=** input**('Please input the password:')** # 用中间变量测试用户信息  
 data **= {'Username': (**self.getmd5**(**username**)**, **u'Unicode')**, **'Password': (**self.getmd5**(**password**)**, **u'Unicode')}** \_file **=** open**('info/user.pkl'**, **'rb')  
 try:** \_temp **=** pickle.load**(**\_file**)  
 finally:** \_file.close**()** # 验证用户信息  
 **if** data **==** \_temp**:** print**('Logging in successful！')  
 return True  
 else:** print**('Error！')  
 return False  
  
 def get\_username(**self**):# 返回用户名  
 return** self.username  
  
 **def get\_password(**self**):# 返回用户密码  
 return** self.password  
  
 **def main\_reg(**self**):# 注册** print**('Start registration')** self.register**()  
  
 def main(**self**):# 登录** print**('Start logging in')** d **=** self.login**()  
 while not** d**:** d **=** self.login**()**# 中文注册登录  
**class RegistrationChinese(**object**):  
  
 def \_\_init\_\_(**self**): # 初始化函数** self.username **= ''# 初始化用户名** self.password **= ''# 初始化用户密码** # 注册  
 **def start(**self**):** username **=** input**('请创建用户名(必须是3-10位英文字母或数字, 不能包含空格或中文):')** # 正则匹配  
 **if** re.match**(r'\w{3,10}'**, username**):** self.username **=** username  
 **else:** print**('Input error!')** # 输入错误则重新输入  
 **while** self.username **!=** username**:** self.start**()** password **=** input**('请创建密码(必须是6-20位英文字母或数字, 不能包含空格或中文):')** # 正则匹配  
 **if** re.match**(r'\w{6,20}'**, password**):** self.password **=** password  
 # 输入错误则重新输入  
 **while** self.password **!=** password**:** self.start**()**

**# 注册  
 def register(**self**):** self.start**()** # MD5加密  
 db**[**self.username**] =** self.getmd5**(**self.password **+** self.username**)** # 将信息通过dict类型格式化二进制存入文件中  
 \_file **=** open**('info/user.pkl'**, **'wb')** data **= {'Username': (**self.getmd5**(**self.username**)**, **u'Unicode')**, **'Password': (**self.getmd5**(**self.password**)**, **u'Unicode')}  
 try:** pickle.dump**(**data, \_file**)  
 finally:** \_file.close**()** print**('创建账户成功！')** # 静态方法求MD5  
 **@**staticmethod  
 **def getmd5(**s**):** md5 **=** hashlib.md5**()** md5.update**(**s.encode**('utf-8'))  
 return** md5.hexdigest**()** # 登录  
 **def login(**self**):** username **=** input**('请输入用户名:')** password **=** input**('请输入密码:')** # 输入错误则重新输入  
 data **= {'Username': (**self.getmd5**(**username**)**, **u'Unicode')**, **'Password': (**self.getmd5**(**password**)**, **u'Unicode')}** \_file **=** open**('info/user.pkl'**, **'rb')  
 try:** \_temp **=** pickle.load**(**\_file**)  
 finally:** \_file.close**()** # 验证用户信息  
 **if** data **==** \_temp**:** print**('登录成功！')  
 return True  
 else:** print**('错误！')  
 return False  
  
 def get\_username(**self**): # 返回用户名  
 return** self.username  
  
 **def get\_password(**self**): # 返回用户密码  
 return** self.password  
  
 **def main\_reg(**self**): # 注册** print**('开始注册')** self.register**()  
  
 def main(**self**): # 登录** print**('开始登录')** b **=** self.login**()  
 while not** b**:** b **=** self.login**()**# 初始化模块  
**class ModuleInit(**object**):  
  
 def \_\_init\_\_(**self**): # 初始化函数  
 pass** # 静态初始化函数  
 **@**staticmethod  
 **def load(**self**): # 加载模块** pygame.init**()** pygame.mixer.init**()** # 错误处理  
 **if not** pygame.mixer**:** print**('Warning, sound disabled!')**# 加载背景  
**class Background(**object**):  
  
 def \_\_init\_\_(**self**): # 初始化函数  
 pass** # 静态方法  
 **@**staticmethod  
 **def load\_image(**self**): # 加载图片** \_screen **=** pygame.display.set\_mode**(**SCREEN\_SIZE**)** # 设置标题  
 pygame.display.set\_caption**('LongPlay')** background **=** pygame.image.load\_extended**(**background\_image\_filename**)**.convert\_alpha**()** disk **=** pygame.image.load\_extended**(**disk\_image\_filename**)**.convert\_alpha**()** stick **=** pygame.image.load\_extended**(**stick\_image\_filename**)**.convert\_alpha**()** bar **=** pygame.image.load\_extended**(**bar\_image\_filename**)**.convert\_alpha**()** # 画出背景  
 \_screen.blit**(**background, **(**0, 0**))** \_screen.blit**(**disk, **(**10, 10**))** \_screen.blit**(**stick, **(**10, 10**))** \_screen.blit**(**bar, **(**350, 415**))** # 刷新页面  
 pygame.display.update**()**# 按钮渲染  
**class Button(**object**):  
  
 def \_\_init\_\_(**self, image\_filename, position**): # 初始化函数** self.position **=** position  
 self.image **=** pygame.image.load\_extended**(**image\_filename**)  
  
 def render(**self, \_surface**): # 渲染按钮** x, y **=** self.position  
 \_w, \_h **=** self.image.get\_size**()** x **-=** \_w **/** 2  
 y **-=** \_h **/** 2  
 \_surface.blit**(**self.image, **(**x, y**))** # 如果point在自身范围内，返回True  
 **def is\_over(**self, point**):** point\_x, point\_y **=** point  
 x, y **=** self.position  
 \_w, \_h **=** self.image.get\_size**()** x **-=** \_w **/** 2  
 y **-=** \_h **/** 2  
 in\_x **=** x **<=** point\_x **<** x **+** \_w  
 in\_y **=** y **<=** point\_y **<** y **+** \_h  
 **return** in\_x **and** in\_y  
  
  
# 喜好功能  
**class Preference(**object**):  
  
 def \_\_init\_\_(**self, image\_filename, position**): # 初始化函数** self.position **=** position  
 self.image **=** pygame.image.load\_extended**(**image\_filename**)  
  
 def render(**self, \_surface**): # 渲染按钮** x, y **=** self.position  
 \_w, \_h **=** self.image.get\_size**()** x **-=** \_w **/** 2  
 y **-=** \_h **/** 2  
 \_surface.blit**(**self.image, **(**x, y**))  
  
 def is\_over(**self, point**):** # 如果point在自身范围内，返回True  
point\_x, point\_y **=** point  
 x, y **=** self.position  
 \_w, \_h **=** self.image.get\_size**()** x **-=** \_w **/** 2  
 y **-=** \_h **/** 2  
 in\_x **=** x **<=** point\_x **<** x **+** \_w  
 in\_y **=** y **<=** point\_y **<** y **+** \_h  
 **return** in\_x **and** in\_y  
  
  
# 音乐播放模式  
**class Mode(**object**):  
  
 def \_\_init\_\_(**self, image\_filename, position**): # 初始化函数** self.position **=** position  
 self.image **=** pygame.image.load\_extended**(**image\_filename**)  
  
 def render(**self, \_surface**): # 渲染按钮** x, y **=** self.position  
 \_w, \_h **=** self.image.get\_size**()** x **-=** \_w **/** 2  
 y **-=** \_h **/** 2  
 \_surface.blit**(**self.image, **(**x, y**))  
  
 def is\_over(**self, point**):** # 如果point在自身范围内，返回True  
point\_x, point\_y **=** point  
 x, y **=** self.position  
 \_w, \_h **=** self.image.get\_size**()** x **-=** \_w **/** 2  
 y **-=** \_h **/** 2  
 in\_x **=** x **<=** point\_x **<** x **+** \_w  
 in\_y **=** y **<=** point\_y **<** y **+** \_h  
 **return** in\_x **and** in\_y  
  
  
# 选项  
**class Options(**object**):  
  
 def \_\_init\_\_(**self, image\_filename, position**): # 初始化函数** self.position **=** position  
 self.image **=** pygame.image.load\_extended**(**image\_filename**)  
  
 def render(**self, \_surface**): # 渲染按钮** x, y **=** self.position  
 \_w, \_h **=** self.image.get\_size**()** x **-=** \_w **/** 2  
 y **-=** \_h **/** 2  
 \_surface.blit**(**self.image, **(**x, y**))  
  
 def is\_over(**self, point**):** # 如果point在自身范围内，返回True  
point\_x, point\_y **=** point  
 x, y **=** self.position  
 \_w, \_h **=** self.image.get\_size**()** x **-=** \_w **/** 2  
 y **-=** \_h **/** 2  
 in\_x **=** x **<=** point\_x **<** x **+** \_w  
 in\_y **=** y **<=** point\_y **<** y **+** \_h  
 **return** in\_x **and** in\_y  
  
  
# 音量设置  
**class VolumeSet(**object**):  
  
 def \_\_init\_\_(**self, image\_filename, position**): # 初始化函数** self.position **=** position  
 self.image **=** pygame.image.load\_extended**(**image\_filename**)  
  
 def render(**self, \_surface**): # 渲染按钮** x, y **=** self.position  
 \_w, \_h **=** self.image.get\_size**()** x **-=** \_w **/** 2  
 y **-=** \_h **/** 2  
 \_surface.blit**(**self.image, **(**x, y**))  
  
 def is\_over(**self, point**):** point\_x, point\_y **=** point  
 x, y **=** self.position  
 \_w, \_h **=** self.image.get\_size**()** x **-=** \_w **/** 2  
 y **-=** \_h **/** 2  
 in\_x **=** x **<=** point\_x **<** x **+** \_w  
 in\_y **=** y **<=** point\_y **<** y **+** \_h  
 **return** in\_x **and** in\_y  
  
  
# 进度条  
**class ProgressBar(**object**):  
  
 def \_\_init\_\_(**self**): # 初始化函数** self.image **=** pygame.image.load\_extended**(**circle\_image\_filename**)** self.initial\_position **= (**348, 413**)  
  
 def render(**self, \_surface**): # 渲染按钮** \_surface.blit**(**self.image, self.initial\_position**)  
  
 def is\_over(**self, point**):** # 如果point在自身范围内，返回True  
point\_x, point\_y **=** point  
 x, y **=** self.initial\_position  
 \_w, \_h **=** self.image.get\_size**()** x **-=** \_w **/** 2  
 y **-=** \_h **/** 2  
 in\_x **=** x **<=** point\_x **<** x **+** \_w  
 in\_y **=** y **<=** point\_y **<** y **+** \_h  
 **return** in\_x **and** in\_y  
  
  
# 编辑  
**class Edit(**object**):  
  
 def \_\_init\_\_(**self, image\_filename, position**): # 初始化函数** self.position **=** position  
 self.image **=** pygame.image.load\_extended**(**image\_filename**)  
  
 def render(**self, \_surface**): # 渲染按钮** x, y **=** self.position  
 \_w, \_h **=** self.image.get\_size**()** x **-=** \_w **/** 2  
 y **-=** \_h **/** 2  
 \_surface.blit**(**self.image, **(**x, y**))  
  
 def is\_over(**self, point**):** # 如果point在自身范围内，返回True  
point\_x, point\_y **=** point  
 x, y **=** self.position  
 \_w, \_h **=** self.image.get\_size**()** x **-=** \_w **/** 2  
 y **-=** \_h **/** 2  
 in\_x **=** x **<=** point\_x **<** x **+** \_w  
 in\_y **=** y **<=** point\_y **<** y **+** \_h  
 **return** in\_x **and** in\_y  
  
  
# 从source文件夹读取音乐文件  
**class GetMusic(**object**):  
  
 def \_\_init\_\_(**self**): # 初始化函数  
 pass** # 静态方法  
 **@**staticmethod  
 **def path(**self, path**): # 获取播放列表** raw\_file\_names **=** os.listdir**(**path**)** music\_files **= []  
 for** file\_name **in** raw\_file\_names**:  
 if** file\_name.lower**()**.endswith**('.mp3') or** file\_name.lower**()**.endswith**('.ogg') or** file\_name.lower**()**.endswith**('.wav'):** music\_files.append**(**os.path.join**(**MUSIC\_PATH, file\_name**))  
 return** sorted**(**music\_files**)**

# 六、模块分析

1. 语言模块

系统将提示用户输入数字选择中英文

英文开始模式提供英文选项

中文开始模式提供中文选项

英文注册登录提供英文的注册登录功能

中文注册登录提供中文的注册登录功能

2. 初始化模块

初始化模块将加载所需的Pygame和mixer模块。

3. 背景加载模块

从指定的路径加载背景图片元素

4. 按钮渲染模块

指定按钮的位置

判断是否点击了按钮

画出控制，喜好，播放模式，选项，音量，进度条，编辑按钮

5. 音乐读取模块

从指定的文件夹读取所有音乐文件

# 七、比较有特色的函数

1. **def is\_over(**self, point**):** point\_x, point\_y **=** point  
 x, y **=** self.position  
 \_w, \_h **=** self.image.get\_size**()** x **-=** \_w **/** 2  
 y **-=** \_h **/** 2  
 in\_x **=** x **<=** point\_x **<** x **+** \_w  
 in\_y **=** y **<=** point\_y **<** y **+** \_h  
 **return** in\_x **and** in\_y

2. # 如果一曲播放结束，就“模拟”按下"next"  
**if** event.type **==** track\_end**:** button\_pressed **= 'next'**

3.# 静态方法  
 **@**staticmethod  
 **def path(**self, path**):** raw\_file\_names **=** os.listdir**(**path**)** music\_files **= []  
 for** file\_name **in** raw\_file\_names**:  
 if** file\_name.lower**()**.endswith**('.mp3') or** file\_name.lower**()**.endswith**('.ogg') or** file\_name.lower**()**.endswith**('.wav'):** music\_files.append**(**os.path.join**(**MUSIC\_PATH, file\_name**))  
 return** sorted**(**music\_files**)**

# 八、存在的不足与对策、编程体会

推荐系统没有真正实现，仅仅具有雏形。需要给每个音乐指定一个标记，一旦按下‘喜爱’的红心，标记就改变值，将此改变记入用户档案中。根据乐曲的风格制作一个数据库，再用LFM（潜在因子算法）得出用户的喜好进行定向推荐并给用户分类。

由于Pygame本身的限制，难以实现平滑的动画效果，与用户的交互效果一般。预想中应实现的背景中唱片旋转和用户拖动唱片效果没有实现。Pygame中的music模块每次都只读取音乐的一部分以提高性能，因此进度条功能没有真正实现。点击红心后的动画效果也没有实现。应学习javascript和数据库等进行实现。

# 九、程序源代码

#!/usr/bin/env python  
# -\*- coding: utf-8 -\*-  
# Author: 张天弈  
  
  
**"""  
Python 3.4.4  
Pygame 1.9.1  
基于Python及其模块Pygame实现的LongPlay音乐播放器  
支持MP3, OGG, WAV格式的音频  
默认随机播放  
可以用上下箭头或点击音量图标控制音量  
可以用左右箭头或点击前后图标切换音乐  
可以用空格键开始或暂停音乐  
按停止键有2.5秒淡出效果  
可添加或删除音乐  
有用户注册及登陆功能  
由正则表达式匹配用户名与密码  
可选择是否注册账户  
使用MD5方法进行数据加密  
"""**# 导入所需模块  
**import** sys  
**import** pygame  
**from** pygame.locals **import \*  
import** os  
**import** os.path  
**import** hashlib  
**from** collections **import** defaultdict  
**import** re  
**import** pickle  
**import** pprint  
**import** shutil  
  
  
# 存放音乐文件的位置  
MUSIC\_PATH **= 'source'**# 屏幕大小  
SCREEN\_SIZE **= (**1000, 625**)**# 背景图片  
background\_image\_filename **= 'image/background.jpg'**disk\_image\_filename **= 'image/disk.png'**stick\_image\_filename **= 'image/stick.png'**bar\_image\_filename **= 'image/bar.png'**circle\_image\_filename **= 'image/circle.png'**# 设置字典初始值为N/A  
db **=** defaultdict**(lambda: 'N/A')**# 语言  
**class Language(**object**):  
  
 def \_\_init\_\_(**self**):  
 pass  
  
 @**staticmethod  
 **def choose(**self**):** num **=** input**('1 English, 2 中文:')  
 return** num  
  
  
# 英文开始模式  
**class Select(**object**):  
  
 def \_\_init\_\_(**self**):  
 pass  
  
 @**staticmethod  
 **def choose(**self**):** num **=** input**('1 to create an account, 2 to log in, 3 to start without logging in:')  
 return** num  
  
  
# 中文开始模式  
**class SelectChinese(**object**):  
  
 def \_\_init\_\_(**self**):  
 pass  
  
 @**staticmethod  
 **def choose(**self**):** num **=** input**('按1创建账户, 按2登陆, 按3不登陆直接启动:')  
 return** num  
  
  
# 英文注册登陆  
**class Registration(**object**):  
  
 def \_\_init\_\_(**self**):** self.username **= ''** self.password **= ''** # 注册  
 **def start(**self**):** username **=** input**(  
 'Please create the username(Must be 3-10 characters or numbers, no Chinese or space allowed):')** # 正则匹配  
 **if** re.match**(r'\w{3,10}'**, username**):** self.username **=** username  
 **else:** print**('Input error!')** # 输入错误则重新输入  
 **while** self.username **!=** username**:** self.start**()** password **=** input**(  
 'Please create the password(Must be 6-20 characters or numbers, no Chinese or space allowed):')** # 正则匹配  
 **if** re.match**(r'\w{6,20}'**, password**):** self.password **=** password  
 # 输入错误则重新输入  
 **while** self.password **!=** password**:** self.start**()** # 静态方法求MD5  
 **@**staticmethod  
 **def getmd5(**s**):** md5 **=** hashlib.md5**()** md5.update**(**s.encode**('utf-8'))  
 return** md5.hexdigest**()  
  
 def register(**self**):** self.start**()** # MD5加密  
 db**[**self.username**] =** self.getmd5**(**self.password **+** self.username**)** # 将信息通过dict类型格式化二进制存入文件中  
 \_file **=** open**('info/user.pkl'**, **'wb')** data **= {'Username': (**self.getmd5**(**self.username**)**, **u'Unicode')**, **'Password': (**self.getmd5**(**self.password**)**, **u'Unicode')}  
 try:** pickle.dump**(**data, \_file**)  
 finally:** \_file.close**()** print**('Registration successful！')** # 登陆  
 **def login(**self**):** username **=** input**('Please input the username:')** password **=** input**('Please input the password:')** # 用中间变量测试用户信息  
 data **= {'Username': (**self.getmd5**(**username**)**, **u'Unicode')**, **'Password': (**self.getmd5**(**password**)**, **u'Unicode')}** \_file **=** open**('info/user.pkl'**, **'rb')  
 try:** \_temp **=** pickle.load**(**\_file**)  
 finally:** \_file.close**()** # 验证用户信息  
 **if** data **==** \_temp**:** print**('Logging in successful！')  
 return True  
 else:** print**('Error！')  
 return False  
  
 def get\_username(**self**):  
 return** self.username  
  
 **def get\_password(**self**):  
 return** self.password  
  
 **def main\_reg(**self**):** print**('Start registration')** self.register**()  
  
 def main(**self**):** print**('Start logging in')** d **=** self.login**()  
 while not** d**:** d **=** self.login**()**# 中文注册登陆  
**class RegistrationChinese(**object**):  
  
 def \_\_init\_\_(**self**):** self.username **= ''** self.password **= ''** # 注册  
 **def start(**self**):** username **=** input**('请创建用户名(必须是3-10位英文字母或数字, 不能包含空格或中文):')** # 正则匹配  
 **if** re.match**(r'\w{3,10}'**, username**):** self.username **=** username  
 **else:** print**('Input error!')** # 输入错误则重新输入  
 **while** self.username **!=** username**:** self.start**()** password **=** input**('请创建密码(必须是6-20位英文字母或数字, 不能包含空格或中文):')** # 正则匹配  
 **if** re.match**(r'\w{6,20}'**, password**):** self.password **=** password  
 # 输入错误则重新输入  
 **while** self.password **!=** password**:** self.start**()  
  
 def register(**self**):** self.start**()** # MD5加密  
 db**[**self.username**] =** self.getmd5**(**self.password **+** self.username**)** # 将信息通过dict类型格式化二进制存入文件中  
 \_file **=** open**('info/user.pkl'**, **'wb')** data **= {'Username': (**self.getmd5**(**self.username**)**, **u'Unicode')**, **'Password': (**self.getmd5**(**self.password**)**, **u'Unicode')}  
 try:** pickle.dump**(**data, \_file**)  
 finally:** \_file.close**()** print**('创建账户成功！')** # 静态方法求MD5  
 **@**staticmethod  
 **def getmd5(**s**):** md5 **=** hashlib.md5**()** md5.update**(**s.encode**('utf-8'))  
 return** md5.hexdigest**()** # 登陆  
 **def login(**self**):** username **=** input**('请输入用户名:')** password **=** input**('请输入密码:')** # 输入错误则重新输入  
 data **= {'Username': (**self.getmd5**(**username**)**, **u'Unicode')**, **'Password': (**self.getmd5**(**password**)**, **u'Unicode')}** \_file **=** open**('info/user.pkl'**, **'rb')  
 try:** \_temp **=** pickle.load**(**\_file**)  
 finally:** \_file.close**()** # 验证用户信息  
 **if** data **==** \_temp**:** print**('登陆成功！')  
 return True  
 else:** print**('错误！')  
 return False  
  
 def get\_username(**self**):  
 return** self.username  
  
 **def get\_password(**self**):  
 return** self.password  
  
 **def main\_reg(**self**):** print**('开始注册')** self.register**()  
  
 def main(**self**):** print**('开始登陆')** b **=** self.login**()  
 while not** b**:** b **=** self.login**()**# 初始化模块  
**class ModuleInit(**object**):  
  
 def \_\_init\_\_(**self**):  
 pass** # 静态初始化函数  
 **@**staticmethod  
 **def load(**self**):** pygame.init**()** pygame.mixer.init**()** # 错误处理  
 **if not** pygame.mixer**:** print**('Warning, sound disabled!')**# 加载背景  
**class Background(**object**):  
  
 def \_\_init\_\_(**self**):  
 pass** # 静态方法  
 **@**staticmethod  
 **def load\_image(**self**):** \_screen **=** pygame.display.set\_mode**(**SCREEN\_SIZE**)** # 设置标题  
 pygame.display.set\_caption**('LongPlay')** background **=** pygame.image.load\_extended**(**background\_image\_filename**)**.convert\_alpha**()** disk **=** pygame.image.load\_extended**(**disk\_image\_filename**)**.convert\_alpha**()** stick **=** pygame.image.load\_extended**(**stick\_image\_filename**)**.convert\_alpha**()** bar **=** pygame.image.load\_extended**(**bar\_image\_filename**)**.convert\_alpha**()** # 画出背景  
 \_screen.blit**(**background, **(**0, 0**))** \_screen.blit**(**disk, **(**10, 10**))** \_screen.blit**(**stick, **(**10, 10**))** \_screen.blit**(**bar, **(**350, 415**))** # 刷新页面  
 pygame.display.update**()**# 按钮渲染  
**class Button(**object**):  
  
 def \_\_init\_\_(**self, image\_filename, position**):** self.position **=** position  
 self.image **=** pygame.image.load\_extended**(**image\_filename**)  
  
 def render(**self, \_surface**):** x, y **=** self.position  
 \_w, \_h **=** self.image.get\_size**()** x **-=** \_w **/** 2  
 y **-=** \_h **/** 2  
 \_surface.blit**(**self.image, **(**x, y**))** # 如果point在自身范围内，返回True  
 **def is\_over(**self, point**):** point\_x, point\_y **=** point  
 x, y **=** self.position  
 \_w, \_h **=** self.image.get\_size**()** x **-=** \_w **/** 2  
 y **-=** \_h **/** 2  
 in\_x **=** x **<=** point\_x **<** x **+** \_w  
 in\_y **=** y **<=** point\_y **<** y **+** \_h  
 **return** in\_x **and** in\_y  
  
  
# 喜好功能  
**class Preference(**object**):  
  
 def \_\_init\_\_(**self, image\_filename, position**):** self.position **=** position  
 self.image **=** pygame.image.load\_extended**(**image\_filename**)  
  
 def render(**self, \_surface**):** x, y **=** self.position  
 \_w, \_h **=** self.image.get\_size**()** x **-=** \_w **/** 2  
 y **-=** \_h **/** 2  
 \_surface.blit**(**self.image, **(**x, y**))  
  
 def is\_over(**self, point**):** point\_x, point\_y **=** point  
 x, y **=** self.position  
 \_w, \_h **=** self.image.get\_size**()** x **-=** \_w **/** 2  
 y **-=** \_h **/** 2  
 in\_x **=** x **<=** point\_x **<** x **+** \_w  
 in\_y **=** y **<=** point\_y **<** y **+** \_h  
 **return** in\_x **and** in\_y  
  
  
# 音乐播放模式  
**class Mode(**object**):  
  
 def \_\_init\_\_(**self, image\_filename, position**):** self.position **=** position  
 self.image **=** pygame.image.load\_extended**(**image\_filename**)  
  
 def render(**self, \_surface**):** x, y **=** self.position  
 \_w, \_h **=** self.image.get\_size**()** x **-=** \_w **/** 2  
 y **-=** \_h **/** 2  
 \_surface.blit**(**self.image, **(**x, y**))  
  
 def is\_over(**self, point**):** point\_x, point\_y **=** point  
 x, y **=** self.position  
 \_w, \_h **=** self.image.get\_size**()** x **-=** \_w **/** 2  
 y **-=** \_h **/** 2  
 in\_x **=** x **<=** point\_x **<** x **+** \_w  
 in\_y **=** y **<=** point\_y **<** y **+** \_h  
 **return** in\_x **and** in\_y  
  
  
# 选项  
**class Options(**object**):  
  
 def \_\_init\_\_(**self, image\_filename, position**):** self.position **=** position  
 self.image **=** pygame.image.load\_extended**(**image\_filename**)  
  
 def render(**self, \_surface**):** x, y **=** self.position  
 \_w, \_h **=** self.image.get\_size**()** x **-=** \_w **/** 2  
 y **-=** \_h **/** 2  
 \_surface.blit**(**self.image, **(**x, y**))  
  
 def is\_over(**self, point**):** point\_x, point\_y **=** point  
 x, y **=** self.position  
 \_w, \_h **=** self.image.get\_size**()** x **-=** \_w **/** 2  
 y **-=** \_h **/** 2  
 in\_x **=** x **<=** point\_x **<** x **+** \_w  
 in\_y **=** y **<=** point\_y **<** y **+** \_h  
 **return** in\_x **and** in\_y  
  
  
# 音量设置  
**class VolumeSet(**object**):  
  
 def \_\_init\_\_(**self, image\_filename, position**):** self.position **=** position  
 self.image **=** pygame.image.load\_extended**(**image\_filename**)  
  
 def render(**self, \_surface**):** x, y **=** self.position  
 \_w, \_h **=** self.image.get\_size**()** x **-=** \_w **/** 2  
 y **-=** \_h **/** 2  
 \_surface.blit**(**self.image, **(**x, y**))  
  
 def is\_over(**self, point**):** point\_x, point\_y **=** point  
 x, y **=** self.position  
 \_w, \_h **=** self.image.get\_size**()** x **-=** \_w **/** 2  
 y **-=** \_h **/** 2  
 in\_x **=** x **<=** point\_x **<** x **+** \_w  
 in\_y **=** y **<=** point\_y **<** y **+** \_h  
 **return** in\_x **and** in\_y  
  
  
# 进度条  
**class ProgressBar(**object**):  
  
 def \_\_init\_\_(**self**):** self.image **=** pygame.image.load\_extended**(**circle\_image\_filename**)** self.initial\_position **= (**348, 413**)  
  
 def render(**self, \_surface**):** \_surface.blit**(**self.image, self.initial\_position**)  
  
 def is\_over(**self, point**):** point\_x, point\_y **=** point  
 x, y **=** self.initial\_position  
 \_w, \_h **=** self.image.get\_size**()** x **-=** \_w **/** 2  
 y **-=** \_h **/** 2  
 in\_x **=** x **<=** point\_x **<** x **+** \_w  
 in\_y **=** y **<=** point\_y **<** y **+** \_h  
 **return** in\_x **and** in\_y  
  
  
**class Edit(**object**):  
  
 def \_\_init\_\_(**self, image\_filename, position**):** self.position **=** position  
 self.image **=** pygame.image.load\_extended**(**image\_filename**)  
  
 def render(**self, \_surface**):** x, y **=** self.position  
 \_w, \_h **=** self.image.get\_size**()** x **-=** \_w **/** 2  
 y **-=** \_h **/** 2  
 \_surface.blit**(**self.image, **(**x, y**))  
  
 def is\_over(**self, point**):** point\_x, point\_y **=** point  
 x, y **=** self.position  
 \_w, \_h **=** self.image.get\_size**()** x **-=** \_w **/** 2  
 y **-=** \_h **/** 2  
 in\_x **=** x **<=** point\_x **<** x **+** \_w  
 in\_y **=** y **<=** point\_y **<** y **+** \_h  
 **return** in\_x **and** in\_y  
  
  
# 从source文件夹读取音乐文件  
**class GetMusic(**object**):  
  
 def \_\_init\_\_(**self**):  
 pass** # 静态方法  
 **@**staticmethod  
 **def path(**self, path**):** raw\_file\_names **=** os.listdir**(**path**)** music\_files **= []  
 for** file\_name **in** raw\_file\_names**:  
 if** file\_name.lower**()**.endswith**('.mp3') or** file\_name.lower**()**.endswith**('.ogg') or** file\_name.lower**()**.endswith**('.wav'):** music\_files.append**(**os.path.join**(**MUSIC\_PATH, file\_name**))  
 return** sorted**(**music\_files**)**# 语言选择  
L **=** Language**()**lan\_number **=** L.choose**(**L**)**# 错误处理  
**try:  
 if** lan\_number **== '1':** # 用户选择  
 S **=** Select**()** number **=** S.choose**(**S**)** R **=** Registration**()** # 错误处理  
 **try:  
 if** number **== '1':** # 用户注册  
 R.main\_reg**()  
 elif** number **== '2':** # 用户登录  
 R.main**()  
 elif** number **== '3':  
 pass  
 except** number **!= '1' and** number **!= '2' and** number **!= '3':** print**('Input Error')** sys.exit**()  
 elif** lan\_number **== '2':** # 用户选择  
 SC **=** SelectChinese**()** number\_c **=** SC.choose**(**SC**)** RC **=** RegistrationChinese**()** # 错误处理  
 **try:  
 if** number\_c **== '1':** # 用户注册  
 RC.main\_reg**()  
 elif** number\_c **== '2':** # 用户登录  
 RC.main**()  
 elif** number\_c **== '3':  
 pass  
 except** number\_c **!= '1' and** number\_c **!= '2' and** number\_c **!= '3':** print**('Input Error')** sys.exit**()  
except** lan\_number **!= '1' and** lan\_number **!= '2':** print**('Error')** sys.exit**()**# 初始化  
MI **=** ModuleInit**()**MI.load**(**MI**)**# 坐标  
x1 **=** 210  
y1 **=** 480  
button\_width **=** 150  
  
# 控制按钮  
buttons **= {}  
pass**buttons**['prev'] =** Button**('image/prev.png'**, **(**x1, y1**))**buttons**['pause'] =** Button**('image/pause.png'**, **(**x1 **+** button\_width **\*** 1, y1**))**buttons**['stop'] =** Button**('image/stop.png'**, **(**x1 **+** button\_width **\*** 2, y1**))**buttons**['play'] =** Button**('image/play.png'**, **(**x1 **+** button\_width **\*** 3, y1**))**buttons**['next'] =** Button**('image/next.png'**, **(**x1 **+** button\_width **\*** 4, y1**))**# 喜好按钮  
preferences **= {}  
pass**preferences**['dislike'] =** Preference**('image/full\_heart.png'**, **(**50, 550**))**# 播放模式按钮  
modes **= {}  
pass**modes**['loop'] =** Mode**('image/loop.png'**, **(**912, 550**))**modes**['shuffle'] =** Mode**('image/shuffle.png'**, **(**942, 550**))**modes**['repeat'] =** Mode**('image/repeat.png'**, **(**970, 550**))**# 选项按钮  
options **= {}  
pass**options**['option'] =** Options**('image/option.png'**, **(**970, 30**))**# 音量按钮  
volumes **= {}  
pass**volumes**['up'] =** VolumeSet**('image/volume\_up.png'**, **(**942, 600**))**volumes**['down'] =** VolumeSet**('image/volume\_down.png'**, **(**970, 600**))**# 编辑按钮  
edits **= {}  
pass**edits**['plus'] =** Edit**('image/plus.png'**, **(**30, 30**))**edits**['minus'] =** Edit**('image/minus.png'**, **(**60, 30**))**# 加载音乐  
GM **=** GetMusic**()**music\_filenames **=** GM.path**(**GM, MUSIC\_PATH**)  
if** len**(**music\_filenames**) ==** 0**:** print**('No music files found in '**, MUSIC\_PATH**)** sys.exit**()**# 设置字体  
font **=** pygame.font.SysFont**('Arial'**, 50, **False)**# 错误处理  
**if not** pygame.font**:** print**('Warning, font disabled!')**label\_surfaces **= []**# 文件名显示  
**for** filename **in** music\_filenames**:** txt **=** os.path.split**(**filename**)[-**1**]** print**('Track:'**, txt**)** txt **=** txt.split**('.')[**0**]** surface **=** font.render**(**txt, **True**, **(**255, 255, 255**))** label\_surfaces.append**(**surface**)**current\_track **=** 0  
max\_tracks **=** len**(**music\_filenames**)**# 加载音乐  
pygame.mixer.music.load**(**music\_filenames**[**current\_track**])**clock **=** pygame.time.Clock**()**# 设置播放状态  
playing **= False**paused **= False**track\_end **=** USEREVENT **+** 1  
pygame.mixer.music.set\_endevent**(**track\_end**)**# 主程序循环  
**while True:** screen **=** pygame.display.set\_mode**(**SCREEN\_SIZE, 0**)** button\_pressed **= None** preference\_pressed **= None** option\_pressed **= None** volume\_pressed **= None** mode\_pressed **= None** edit\_pressed **= None** pressed\_keys **=** pygame.key.get\_pressed**()** # 键盘按键控制  
 **if** pressed\_keys**[**K\_UP**]:** pygame.mixer.music.set\_volume**(**pygame.mixer.music.get\_volume**() +** 0.1**)  
 if** pressed\_keys**[**K\_DOWN**]:** pygame.mixer.music.set\_volume**(**pygame.mixer.music.get\_volume**() -** 0.1**)  
 if** pressed\_keys**[**K\_LEFT**]:** button\_pressed **= 'next'  
 if** pressed\_keys**[**K\_RIGHT**]:** button\_pressed **= 'prev'  
 if** pressed\_keys**[**K\_SPACE**]:  
 if** playing**:** pygame.mixer.music.pause**()** playing **= False** paused **= True  
 elif** paused**:** pygame.mixer.music.unpause**()** playing **= True** paused **= False  
 else:** pygame.mixer.music.play**()** playing **= True** paused **= False** # 监听事件  
 **for** event **in** pygame.event.get**():  
 if** event.type **==** QUIT**:** sys.exit**()  
 if** event.type **==** MOUSEBUTTONDOWN**:** # 判断哪个控制按钮被按下  
 **for** button\_name, button **in** buttons.items**():  
 if** button.is\_over**(**event.pos**):** print**(**button\_name, **'pressed')** button\_pressed **=** button\_name  
 **break** # 判断是否按下喜好按钮  
 **for** preference\_push, preference **in** preferences.items**():  
 if** preference.is\_over**(**event.pos**):** print**(**preference\_push, **'Prefer')** preference\_pressed **=** preference\_push  
 **break** # 判断是否按下设置按钮  
 **for** option\_push, option **in** options.items**():  
 if** option.is\_over**(**event.pos**):** print**(**option\_push, **'Option')** option\_pressed **=** option\_push  
 **break** # 判断是否按下音量按钮  
 **for** volume\_push, volume **in** volumes.items**():  
 if** volume.is\_over**(**event.pos**):** print**(**volume\_push, **'Volume')** volume\_pressed **=** volume\_push  
 **break** # 判断是否切换播放模式  
 **for** mode\_push, mode **in** modes.items**():  
 if** mode.is\_over**(**event.pos**):** print**(**mode\_push, **'Mode')** mode\_pressed **=** mode\_push  
 **break** # 判断是否按下编辑按钮  
 **for** edit\_push, edit **in** edits.items**():  
 if** edit.is\_over**(**event.pos**):** print**(**edit\_push, **'Edit')** edit\_pressed **=** edit\_push  
 **break** # 如果一曲播放结束，就“模拟”按下"next"  
 **if** event.type **==** track\_end**:** button\_pressed **= 'next'  
  
 if** edit\_pressed **is not None:  
 if** edit\_pressed **== 'plus':** place **=** input**('Please input the address of the file:')** shutil.copy**(**place, **'source')  
 elif** edit\_pressed **== 'minus':** pygame.mixer.music.stop**()** os.remove**(**music\_filenames**[**current\_track**])  
  
 if** volume\_pressed **is not None:  
 if** volume\_pressed **== 'up':** pygame.mixer.music.set\_volume**(**pygame.mixer.music.get\_volume**() +** 0.1**)  
 elif** volume\_pressed **== 'down':** pygame.mixer.music.set\_volume**(**pygame.mixer.music.get\_volume**() -** 0.1**)  
  
 if** option\_pressed **is not None:** file **=** open**('info/user.pkl'**, **'rb')  
 try:** temp **=** pickle.load**(**file**)** pprint.pprint**(**temp**)  
 finally:** file.close**()  
  
 if** mode\_pressed **is not None:  
 if** mode\_pressed **== 'loop':** current\_pos **=** pygame.mixer.music.get\_pos**()** pygame.mixer.music.play**(**0, current\_pos**)  
 if** mode\_pressed **== 'shuffle':** current\_pos **=** pygame.mixer.music.get\_pos**()** pygame.mixer.music.play**(**0, current\_pos**)  
 if** mode\_pressed **== 'repeat':** current\_pos **=** pygame.mixer.music.get\_pos**()** pygame.mixer.music.play**(**99, current\_pos**)  
  
 if** preference\_pressed **is not None:  
 if** preference\_pressed **== 'dislike':** preferences**['like'] =** Preference**('image/full\_heart.png'**, **(**50, 550**))** preferences**['like']**.render**(**screen**)  
 if** preference\_pressed **== 'like':** # 加载背景  
 \_BG **=** Background**()** \_BG.load\_image**(**\_BG**)** # 写当前歌名  
 \_label **=** label\_surfaces**[**current\_track**]** w1, h1 **=** \_label.get\_size**()** screen\_w1 **=** SCREEN\_SIZE**[**0**]** screen.blit**(**\_label, **((**screen\_w1 **-** w1**) /** 2 **+** 10, 520**))** # 画控制按钮  
 **for** button **in** buttons.values**():** button.render**(**screen**)** # 画喜好按钮  
 **for** preference **in** preferences.values**():** preferences**['dislike']**.render**(**screen**)** # 画播放模式按钮  
 **for** mode **in** modes.values**():** mode.render**(**screen**)** # 画选项按钮  
 **for** option **in** options.values**():** option.render**(**screen**)** # 画音量按钮  
 **for** volume **in** volumes.values**():** volume.render**(**screen**)** # 帧率设置  
 clock.tick**(**50**)** # 页面刷新  
 pygame.display.update**()  
  
 if** button\_pressed **is not None:  
 if** button\_pressed **== 'next':** current\_track **= (**current\_track **+** 1**) %** max\_tracks  
 pygame.mixer.music.load**(**music\_filenames**[**current\_track**])  
 if** playing**:** pygame.mixer.music.play**()  
  
 elif** button\_pressed **== 'prev':** # 按下prev的逻辑：  
 # 如果已经播放音乐超过了3秒，就从头开始，否则就播放上一首  
 **if** pygame.mixer.music.get\_pos**() >** 3000**:** pygame.mixer.music.stop**()** pygame.mixer.music.play**()  
 else:** current\_track **= (**current\_track **-** 1**) %** max\_tracks  
 pygame.mixer.music.load**(**music\_filenames**[**current\_track**])  
 if** playing**:** pygame.mixer.music.play**()  
  
 elif** button\_pressed **== 'pause':  
 if** paused**:** pygame.mixer.music.unpause**()** paused **= False  
 else:** pygame.mixer.music.pause**()** paused **= True  
  
 elif** button\_pressed **== 'stop':** # 淡出效果  
 pygame.mixer.music.fadeout**(**2500**)** playing **= False  
  
 elif** button\_pressed **== 'play':  
 if** paused**:** pygame.mixer.music.unpause**()** paused **= False  
 else:  
 if not** playing**:** pygame.mixer.music.play**()** playing **= True** # 加载背景  
 BG **=** Background**()** BG.load\_image**(**BG**)** # 写当前歌名  
 label **=** label\_surfaces**[**current\_track**]** w, h **=** label.get\_size**()** screen\_w **=** SCREEN\_SIZE**[**0**]** screen.blit**(**label, **((**screen\_w **-** w**) /** 2 **+** 10, 520**))** # 画控制按钮  
 **for** button **in** buttons.values**():** button.render**(**screen**)** # 画喜好按钮  
 **for** preference **in** preferences.values**():** preferences**['dislike']**.render**(**screen**)** # 画播放模式按钮  
 **for** mode **in** modes.values**():** mode.render**(**screen**)** # 画选项按钮  
 **for** option **in** options.values**():** option.render**(**screen**)** # 画音量按钮  
 **for** volume **in** volumes.values**():** volume.render**(**screen**)** # 画编辑按钮  
 **for** edit **in** edits.values**():** edit.render**(**screen**)** # 画进度条  
 PB **=** ProgressBar**()** PB.render**(**screen**)** # 帧率设置  
 clock.tick**(**50**)** # 页面刷新  
 pygame.display.update**()**