**物联网移动应用基础**

**课程设计说明书**

**（20 20 ～ 20 21 学年第二学期）**

**题目名称： 视频播放器**

**姓 名 谢晓艳**

**学 号 190611243**

**专 业 物联网工程**

**班 级 物联19102**

**指导教师 李剑波、丁德红**

**提交日期 2021.6.12**

**湖南应用技术学院课程设计任务书**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 课程名称 | 物联网移动应用基础课程设计 | | | | 课程代码 | | 190809050109 |
| 设计题目 | 视频播放器 | | | | 分组人数 | | 1人 |
| 设计时间 | 2021年6月7日—2021年6月13日 | | | | | | |
| 学院 | 信息工程学院 | 专业 | 物联网工程 | 班级 | | 物联19102 | |
| **一、课程设计任务（条件）、开发工具及运行环境**  1、设计任务  （1）主界面：上面为播放界面，下面的视频控制条作为弹出窗口可以显示或隐藏。  （2）用户可以通过视频控制条来控制视频的播放，包括视频进度的控制、视频源的选择、播放暂停功能的转换、上一个视频和下一个视频的选择、音量大小的控制等等。  （3）用户打开视频播放器时，可以通过左下角的“弹出”按钮，显示当前SD卡目录中的所有满足条件的视频文件，从中选择想要播放的视频。  （4）设计音量控制界面，当用户单击控制条中的音量图标时将会弹出音量控制界面。通过上下触摸可以增大或减小音量。  （5）能够设置视频播放的尺寸是否为全屏。  （6）能够获取视频的播放位置并开始播放。  2、开发工具及运行环境   1. 开发工具：Android Studio、AndroidSDK和JDK 2. 运行环境：Windows10 | | | | | | | |
| **二、对课程设计成果的要求（包括课程设计说明书、图纸、图表、实物等软硬件要求）**  1、设计说明书要求  内容完整，概念清楚，方案合理，数据可靠，图文（表）规范、排版工整。  2、设计系统的要求   1. 实现全部功能 2. 执行效率高 3. 界面友好、交互性强   3、编码的要求   1. 声明   成员声明变量：在类的第一个方法之前声明变量  变量声明：代码块开始处声明变量  类和接口声明：在类定义、接口定义或方法定义中，“{”在声明部分的最右侧，“}”独占一行，需要与相应的声明语句对齐。若为空语句，“}”应紧跟在“{”之后   1. 命名规范   类和接口命名：采用驼峰式命名规则，要求第一个单词首字母大写  方法命名：采用驼峰式命名规则，要求第一个单词的首字母小写  变量命名：每个字母全部小写，可用下划线隔开  常量命名：每个字母全部大写，可用下划线隔开   1. 缩进   代码以1个Tab键或4个空格缩进 | | | | | | | |
| **三、课程设计工作进度计划**  2021年5月25-28日：完成选题。  2021年6月5日：完成《课程设计说明书》的封面、表格内容填写；完成需求分析、概要设计。填写《课程设计说明书》的“一、需求分析”、“二、概要设计”内容的填写。  2021年6月6日：详细设计。绘制模块功能表；确定每个模块的算法、流程图、和模块中各数据说明表；确定各模块的GUI界面。  2021年6月7-11日：代码编写。  2021年6月12日：系统测试；完成《课程设计说明书》中“三、详细设计”内容。  2021年6月13日：完善程序和课程设计说明书、答辩。  2021年6月14日：根据答辩过程指导的意见，对系统和《课程设计说明书》进行修改，提交项目源代码和《课程设计说明书》（.doc文件） | | | | | | | |
| 四、主要参考资料  [1] 郭志宏. Android应用开发详解[M].电子工业出版社. 2010.  [2] 杨丰盛.Android应用开发揭秘[M]. 机械工业出版社. 2010.  [3] 余志龙，陈昱勋，郑名杰，陈小凤，郭秩均. Google Android SDK开发范例 大全[M].人民邮电出版社. 2009.  [4] 李宁. Android/OPhone开发完全讲义[M]. 中国水利水电出版社. 2010.  [5] Ed Burnett, Hello, Android: Introducing Google’s Mobile Development Platform[J]. PRAGMATIC BOOKSHELF,2010.7:10-11.  [6] Rick Rogers, John Lombardo.Android Application Development[M].O’Reilly Media,Inc.2009.  [7] 胡伟.Android 系统架构及其驱动研究[J].广州广播电视大学学报. 2010,10(4).  [8] 靳岩，姚尚朗. Google Android开发入门与实践[M].人民邮电出版社. 2009.  [9] 陈建伟. 基于Android的三维物体的触摸控制[J].硅谷,2009,(23):76.  [10] E2ECloud工作室. 深入浅出Google Android[M].人民邮电出版社. 2009. | | | | | | | |

指导教师（签名）： 教研室主任（签名）：

年 月 日 年 月 日

**目录**

[一、 需求分析 1](#_Toc25869)

[1.1 引言 1](#_Toc21997)

[1.2 功能结构 1](#_Toc5883)

[二、 总体设计 2](#_Toc7986)

[2.1 系统设计基本思路 2](#_Toc20755)

[2.2 系统结构设计 2](#_Toc5584)

[2.3 系统流程设计 3](#_Toc19116)

[2.4 系统开发目录 3](#_Toc11341)

[三、 详细设计 4](#_Toc10876)

[3.1 难点分析 4](#_Toc7083)

[3.2 基础类分析 4](#_Toc17939)

[3.3 闪屏界面分析 6](#_Toc18799)

[3.4 主界面分析 7](#_Toc595)

[3.5 音频播放界面分析 8](#_Toc28335)

[3.6 视频播放界面分析 10](#_Toc1197)

[四、 程序运行结果测试与分析 13](#_Toc4338)

[4.1 闪屏界面测试 13](#_Toc8988)

[4.2 主界面测试 13](#_Toc8862)

[4.3 音频播放界面测试 14](#_Toc12610)

[4.4 视频播放界面测试 15](#_Toc4170)

[五、 结论与心得 16](#_Toc31498)

[六、 附：代码 17](#_Toc26928)

**课程设计说明书**

# 需求分析

## 引言

随着物质水平的提高，人们开始更加注重精神的享受。目前社会上多媒体播放器系统发展飞快，视频音频播放软件在人们的日常生活和学习、科研、生产等各个方面有着重要的应用。因此开发一款实用的多媒体播放器是很必要的。此次设计拟开发一个简易多媒体播放器，以提高对所学知识的掌握程度，并为开发功能更强大的播放器积累经验。

对于视频播放应当具有基本的播放暂停、调节进度、全屏缩屏、调节音量、调节屏幕亮度等功能。对于音频播放应当具有基本的播放暂停、调节进度、显示播放列表、播放模式选择、显示歌词等功能。通过根据现有的知识对软件所需的功能进行评估分析，编写算法最终编写出此程序。

本项目是一款基于Android手机平台的视频音频播放器，使Android手机拥有个性的多媒体播放器，使手机显得更生动灵活化，与人们更为接近，让手机主人随时随地看视频听音乐。使人们的生活更加多样化，也使设计者更加熟悉Android的技术和其它在市场上的特点。

## 功能结构

功能分析：

1. 实现手机支持的视频、音频格式文件的播放功能
2. 能在手机里定位视频、音频文件
3. 能智能搜索所有视频、音频文件
4. 能对视频、音频文件常规信息进行读取

视频功能结构如图1.1所示，音频功能结构如图1.2所示

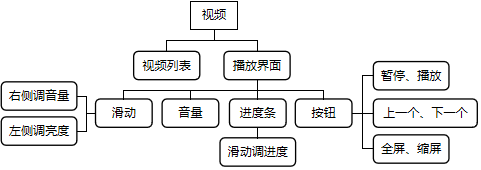


图 1.1 视频功能结构图

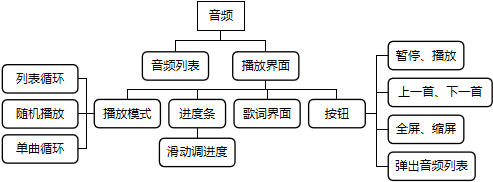


图 1.2 音频功能结构图

# 总体设计

## 系统设计基本思路

第一步：程序运行通过闪屏界面后进入主Activity，通过ViewPager可以在两个不同的Fragment切换。

第二步：设计视频、音频文件列表，读取视频、音频文件信息，将所有视频、音频文件列表显示。

第三步：设计音频播放界面，实现基本的音频功能

第四步；设计视频播放界面，实现基本的视频功能

## 系统结构设计

播放器应用程序有5个Activity，分别是BaseActivity、SplashActivity、MainActivity、AudioPlayerActivity、VideoPlayerActivity。每一个Activity实现相应部分的功能，每一个Activity对应一个或多个.xml的布局文件。在布局文件中，添加用来实现相应部分功能的各种控件。

BaseActivity是其他4个Activity的基类，其他activity应该继承这个类。BaseActivity主要功能是权限的检查和申请，使用了EasyPermissions库。EasyPermissions库是一个方便开发者为App高效处理危险权限的库。Android现在将所有的权限归为了两类，一类是普通权限，一类是危险权限。对于普通权限，系统会自动帮我们进行授权，不需要手动操作。对于危险权限，必须要由用户手动进行点击授权才可以，否则程序无法完成相应的功能。

SplashActivity的主要功能是闪屏，在MainActivity通过Intent进入闪屏页。

MainActivity是主活动，布局文件是activity\_main.xml文件，因为是主界面，所以退出时需要按两次返回按钮。在该活动中，需要实现AudioFragment和VideoFragment的切换，实现指示线的滚动、标题状态的改变。

AudioPlayerActivity主要实现音频播放功能，控件数量多，功能繁多。存在TextView控件是实现显示歌名、显示歌手、显示时长；存在Button控件实现返回主界面、播放模式选择、上一首、播放、暂停、下一首、播放列表弹出；存在ImageView控件实现动态视图；存在SeekBar控件实现进度控制；存在LyricView控件实现歌词滚动显示。

VideoPlayerActivity主要实现视频播放功能，控件数量多，功能繁多。存在TextView控件是实现显示视频名、显示电量、显示当前时间、显示时长；存在Button控件实现是否静音、返回主界面、上一首、播放、暂停、下一首、是否全屏；存在IMyVideoView控件实现视频播放；存在SeekBar控件实现进度控制、音量控制。

## 系统流程设计

用户启动程序，将进入软件的闪屏界面等待5秒后，进入主界面，主界面会判断是否进入音频列表界面，是则进入音频列表界面，否则进入视频列表界面。如果进入音频列表界面，用户可以看到所有音频文件列表显示，点击可进入音频播放界面；如果进入视频列表界面，用户可以看到所有视频文件列表显示，点击可进入视频播放界面。系统流程如图2.1所示

在音频播放界面，可以播放音频、显示歌词、显示播放列表、选择播放模式。

在视频播放界面，可以播放视频、调亮度、调音量、全屏缩屏。

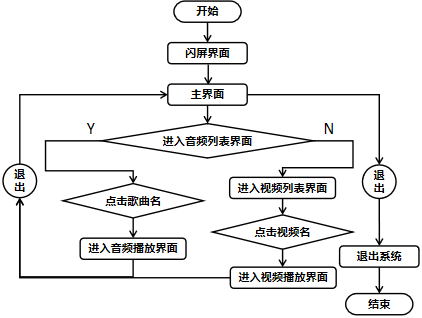
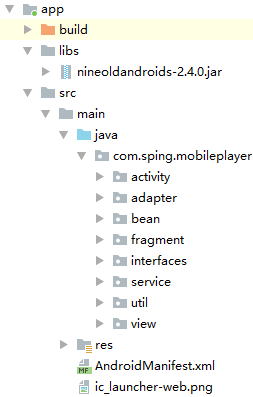


图 2.1 系统流程图

## 系统开发目录

1. 项目开发目录

src：源代码文件

res：目录存放Android项目的各种资源文件

libs：存储Android项目开发所使用的第三方jar包

build：Android项目自动生成的各种源文件，R.java文件存放在此文件中

1. 包

activity：活动

adapter：适配器

bean：存放实体类

fragment：碎片

interfaces：接口

service：服务

util：通用

view：界面

1. 系统开发目录，如图2.2所示

# 详细设计

图 2.2 系统开发目录图

## 难点分析

1. 通过使用android.support.v4.view.ViewPager使得两个Fragment能滑动切换，在Fragment中通过使用自定义的适配器AudioListAdapterRV来填充RecyclerView中的数据。
2. 在使用RecyclerView的时候，我们将需要使用到cursorAdapter，但当我们使用RecyclerView的时候，Adapter为cursorAdapter是会报错的，到目前为止，RecyclerView是不支持cursorAdapter的。
3. 如何使用Cursor设置过滤条件，使得在RecyclerView显示我们需要的信息列表。

## 基础类分析

1. Fragment的基类

表 1 BaseFragment成员函数表

|  |  |
| --- | --- |
| **成员函数** | **说明** |
| View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) | 方法重写，创建View |
| <T> T findView(int id) | 查找View ,省略强转操作 |
| void showToast(String text) | 在屏幕中间显示一个Toast |
| void onClick(View v) | 方法重写，点击 |

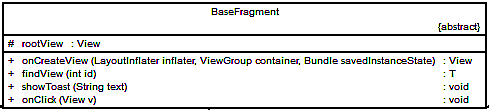


图 3.1 BaseFragment类的UML图

1. activity 的基类

表 2 BaseActivity成员函数表

|  |  |
| --- | --- |
| **成员函数** | **说明** |
| void onCreate(Bundle savedInstanceState) | 方法重写，创建 |
| <T> T findView(int id) | 减少findViewById的使用 |
| void onClick(View v) | 方法重写，点击 |
| void onActivityResult(int requestCode, int resultCode, Intent data) | 方法重写，从软件设置界面，返回当前程序时候 |
| boolean checkPermission(Activity context, String[] perms) | 检查权限 |
| void requestPermission(Activity context, String tip, int requestCode, String[] perms) | 请求权限 |
| void onPermissionsGranted(int requestCode, List<String> perms) | 请求权限成功 |
| void onPermissionsDenied(int requestCode, List<String> perms) | 请求权限失败 |
| onRequestPermissionsResult(int requestCode, String[] permissions, int[] grantResults) | 重写onRequestPermissionsResult，用于接受请求结果 |

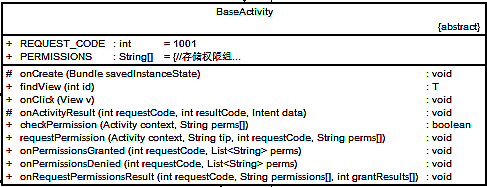


图 3.2 BaseActivity类的UML图

1. Utils类，通用

表 3 Utils成员函数表

|  |  |
| --- | --- |
| **成员函数** | **说明** |
| void setButtonOnClickListener(View view, OnClickListener listener) | 查找Button/ImageButtom并设置单击监听器 |
| void showToast(Context context, String text) | 在屏幕中间显示一个Toast |
| int getScreenWidth(Context context) | 获取屏幕宽度 |
| int getScreenHeight(Context context) | 获取屏幕高度 |
| void printCursor(Cursor cursor) | 遍历Cursor中的所有记录 |
| CharSequence formatMillis(long duration) | 格式化一个毫秒值 |

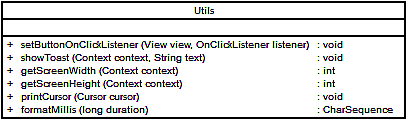


图 3.3 Utils类的UML图

## 闪屏界面分析

SplashActivity继承BaseActivity，主要实现延迟0.5秒进入主页，并且在闪屏界面触屏将会直接进入主界面。

主要实现代码：（SplashActivity类图如图3.1所示）

private void delayEnterHome() {//延迟0.5秒进入主页

mHandler = new Handler();

mHandler.postDelayed(new Runnable() {

@Override

public void run() {

enterHome();

}

}, 2000);

}

//进入首页

protected void enterHome() {finish();}

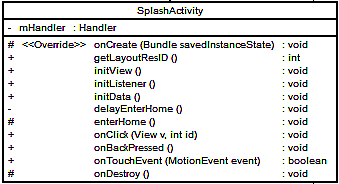


图 3.4 SplashActivity类的UML图

## 主界面分析

1. 主界面布局文件分析

主界面的布局文件activity\_main.xml中，顶部的两个TextView主要是进行界面切换时进行提示，View是指示线，告诉用户现在是哪个界面，ViewPager是进行界面切换的主要实现者。

activity\_main.xml控件分布如图3.2所示

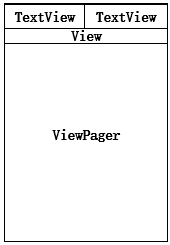


图 3.5 activity\_main.xml控件分布

1. 主界面活动分析

表 5 MainActivity成员变量表

|  |  |  |
| --- | --- | --- |
| **字段名** | **数据类型** | **说明** |
| isExit | boolean | 是否退出的标志 |
| handler | Handler | 接收用户发送信息的handler |
| tv\_video | TextView | 视频 |
| tv\_audio | TextView | 音频 |
| view\_pager | ViewPager | 视图翻页工具，提供了多页面切换的效果 |
| view\_indicator | View | 指示线 |
| indicatorWidth | int | 指示线宽度 |

表 6 MainActivity成员函数表

|  |  |
| --- | --- |
| **成员函数** | **说明** |
| onKeyDown(int keyCode, KeyEvent event) | 监听手机物理按键单击事件  （退出时需要按两次返回按钮） |
| onCreate(Bundle savedInstanceState) | 进入闪屏界面 |
| getLayoutResID() | 返回一个用于显示界面的布局ID |
| initView() | 初始化View |
| initListener() | 初始化监听器 |
| initData() | 初始化数据 |
| onClick(View v, int id) | 单击事件 |
| changeTitleTextState(boolean isSelectVideo) | 改变标题状态 |
| scaleTitle(float scale, TextView textViw) | 缩放标题 |

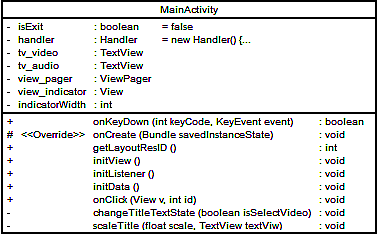


图 3.6 MainActivity的UML图

## 音频播放界面分析

1. 音频播放界面布局分析

AudioPlayerActivity主要实现音频播放功能，控件数量多，功能繁多。存在TextView控件是实现显示歌名、显示歌手、显示时长；存在Button控件实现返回主界面、播放模式选择、上一首、播放、暂停、下一首、播放列表弹出；存在ImageView控件实现动态视图；存在SeekBar控件实现进度控制；存在LyricView控件实现歌词滚动显示。

AudioPlayerActivity控件分布如图3.4所示

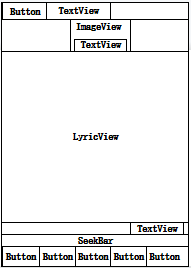


图 3.7 AudioPlayerActivity控件分布图

1. 音频播放功能

视频的信息列表显示，但在使用listview的时候，可能会使用到cursorAdapter。

listiew.setAdapter(cursorAdapter) 是常见的用法。

但是，当我们使用RecyclerView的时候，Adapter为cursorAdapter是会报错的。到目前为止，RecyclerView是不支持cursorAdapter的，但在github上有网友通过改写，已经实现了支持。

难点：数据传输、服务连接、按钮控件功能实现、弹出列表功能实现、歌词界面实现、歌词滚动实现、动态视图实现。

主要代码实现：

protected void startAudioPlayerActivity(ArrayList<AudioItem> audios, int position) {

Intent intent = new Intent(mContext, AudioPlayerActivity.class);

intent.putExtra(Keys.ITEMS, audios);

intent.putExtra(Keys.CURRENT\_POSITION\_IN\_LIST, position);

mContext.startActivity(intent);

}

protected ArrayList<AudioItem> getAudios(Cursor cursor) {

ArrayList<AudioItem> audios = new ArrayList<AudioItem>();

cursor.moveToFirst();

do {

audios.add(AudioItem.fromCursor(cursor));

} while (cursor.moveToNext());

return audios;

}

1. AudioItem设计

该类是音频的实体类，主要是为了能存放音频的信息，并且把一个Cursor对象封装成一个AudioItem对象。

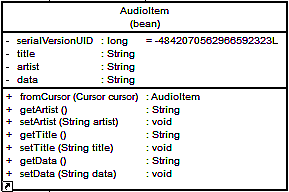


图3.8 AudioItem类的UML图

1. AudioFragment设计

查询数据，进行导入

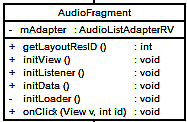


图3.9 AudioFragment类的UML图

1. AudioListAdapter设计

CursorAdapter类是继承于BaseAdapter，是一个虚类，为Cursor和ListView连接提供了桥梁。CursorAdapter是继承了BaseAdapter后，覆盖了getView方法，在getView方法中调用了newView和bindView方法

使用CursorAdapter时必须实现它的两个方法

(1)newView：不是每次都被调用，只在实例化的时候调用，数据增加的时候也会调用，但是在重绘(比如修改条目里的TextView的内容)的时候不会被调用

(2)bindView：在绘制Item之前一定会调用bindView方法，在重绘的时候也同样被调用

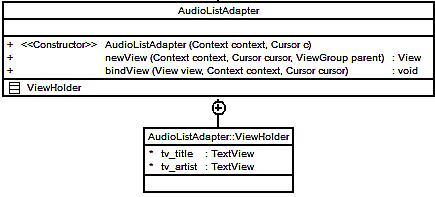


图3.10 AudioListAdapter类的UML图

## 视频播放界面分析

1. 视频播放界面布局分析

VideoPlayerActivity主要实现视频播放功能，控件数量多，功能繁多。存在TextView控件是实现显示视频名、显示电量、显示当前时间、显示时长；存在Button控件实现是否静音、返回主界面、上一首、播放、暂停、下一首、是否全屏；存在IMyVideoView控件实现视频播放；存在SeekBar控件实现进度控制、音量控制。

VideoPlayerActivity控件分布如图2.9所示

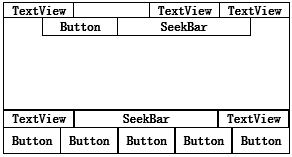


图 2.11 VideoPlayerActivity控件分布图

1. 视频播放功能难点

视频的信息列表显示，但在使用listview的时候，可能会使用到cursorAdapter。

listiew.setAdapter(cursorAdapter) 是常见的用法。

但是，当我们使用RecyclerView的时候，Adapter为cursorAdapter是会报错的。到目前为止，RecyclerView是不支持cursorAdapter的，但在github上有网友通过改写，已经实现了支持。

难点：数据传输、手机电量显示、当前系统时间显示、全屏缩放功能实现。

主要代码实现：

private void startVideoPlayerActivity(ArrayList<VideoItem> videos, int position) {

Intent intent = new Intent(mContext, VideoPlayerActivity.class);

intent.putExtra(Keys.ITEMS, videos);

intent.putExtra(Keys.CURRENT\_POSITION\_IN\_LIST, position);

mContext.startActivity(intent);

}

private ArrayList<VideoItem> getVideoItems(Cursor cursor) {

ArrayList<VideoItem> videos = new ArrayList<VideoItem>();

cursor.moveToFirst();

do {

videos.add(VideoItem.fromCursor(cursor));

} while (cursor.moveToNext());

return videos;

}

1. VideoItem设计

该类是视频的实体类，主要是为了能存放视频的信息，并且把一个Cursor对象封装成一个VideoItem对象。

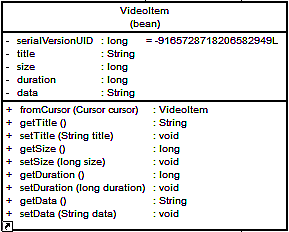


图3.12 VideoItem类的UML图

1. VideoFragment设计

查询数据，进行导入

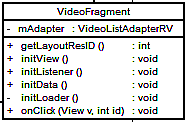


图3.13 VideoFragment类的UML图

1. VideoListAdapter设计

CursorAdapter类是继承于BaseAdapter，是一个虚类，为Cursor和ListView连接提供了桥梁。CursorAdapter是继承了BaseAdapter后，覆盖了getView方法，在getView方法中调用了newView和bindView方法。

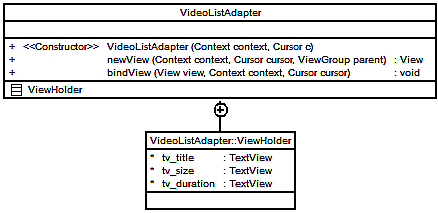


图3.14 VideoListAdapter类的UML图

# 程序运行结果测试与分析

## 闪屏界面测试

在用户进行了存储权限的授权后，将进入闪屏界面，闪屏界面将会缓慢显示一行“好音乐，好心情”。在等待大约5秒后，将会进入主界面。



图4.1 闪屏功能测试图

## 主界面测试

主界面通过Fragment以及ViewPager能够使得用户能在两个Fragment中进行切换，在切换到哪一个时，下方将显示一条横线，提示用户处于哪个界面，以便用户能更好的使用，提高交互性。

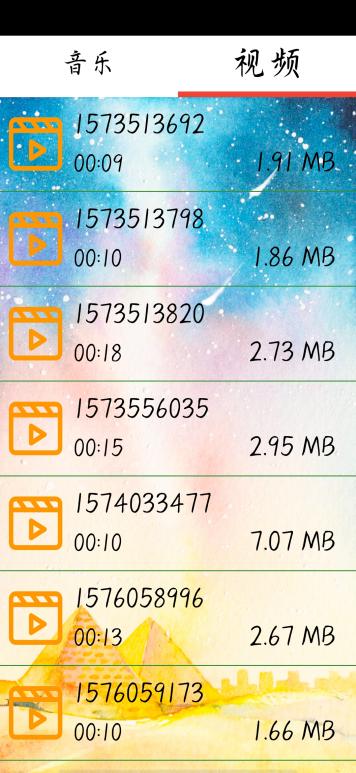
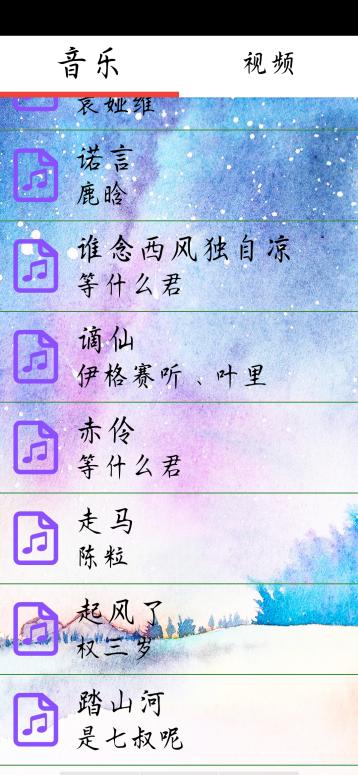


图4.2 主界面功能测试图

## 音频播放界面测试

在主界面的音乐列表中点击选中一首音频，将获取该音频的信息，并显示在相应位置。同时根据显示的图片，存在返回主界面、播放模式选择、上一首、播放、暂停、下一首、播放列表弹出等功能。

当用户开启了通知权限时，将会在通知栏显示自定义的播放控件。



图4.3 音频播放界面测试图



图4.4 通知栏测试图

## 视频播放界面测试

在主界面的视频列表中点击选中一个视频，将获取该视频的信息，并显示在相应位置。同时根据显示的图片，存在是否静音、返回主界面、上一首、播放、暂停、下一首、是否全屏等功能。

在点击视频界面时，将会弹出控件或隐藏控件，还有一个难以显示的功能，当用户在左侧上下滑动时将可以调节视频亮度，当用户在右侧上下滑动时将可以调节视频音量。



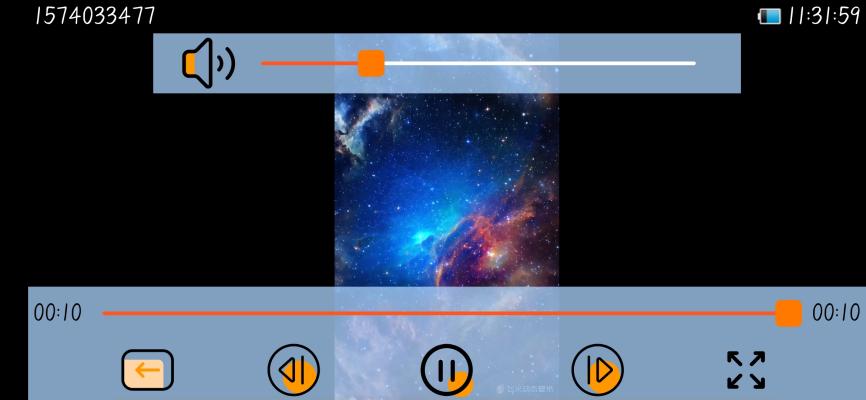
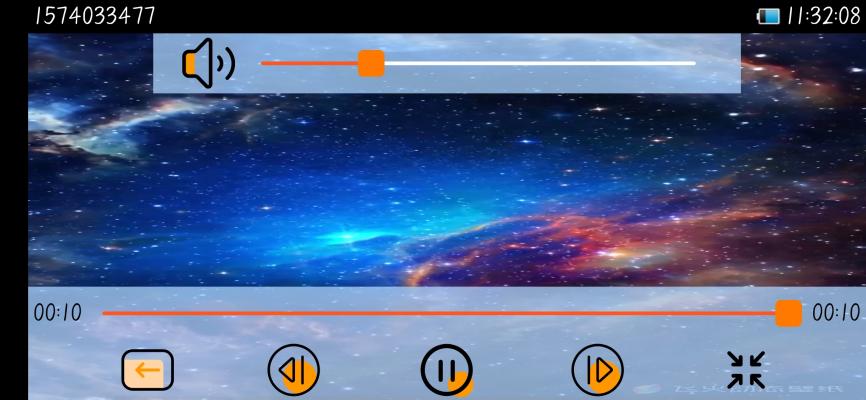


图 4.5 视频播放界面测试图

# 结论与心得

通过这次课程设计，发现其中有很多知识没有接触过，上网查资料的时候才发现自己学到的仅仅只是皮毛，还有很多需要我们掌握的东西不太清楚。也有许多已经学过的知识点没有理解到位，不能灵活的运用于实际，不能很好的用来解决问题。

通过对Android手机平台的音频视频播放器软件的开发，使我对Android播放器系统的整体设计有一个深入的了解，对整个流程也会有一个清晰的认识。开发Android播放器，要抓住开发的核心部分，播放器大体由播放主界面、播放列表、播放设置、文件信息浏览、搜索文件五大核心组成，只要掌握了这五部分的开发，播放器就能初具规模。而其它的功能都是在这六个功能的基础上去补充添加的，但是这是功能的确是必不可少的功能，否则就不能算是播放器了。

音频播放器系统实现了播放器的基本功能：播放，暂停、停止、上/下一首、歌词显示、播放模式、播放列表弹出等功能。视频播放器系统实现了播放器的基本功能：播放，暂停、停止、上/下一首、音量调节、亮度调节、全屏缩屏、手机电量显示、系统时间显示等功能。

总之，经过几个月的Android系统技术知识的学习和对播放器的构思与设计，使我对Android系统有了一定的了解，让我对Android系统上的开发流程和开发模式有了深入的了解，这对我以后的Android项目开发有很大的帮助。

回顾此次课程设计，从理论到实践，这些日子学到了很多知识。只有理论知识是远远不够的，只有把所学的理论知识与实践结合起来，从而提高自己的实际动手能力和独立思考能力。在课程设计的过程中也有遇到许多问题，可以说是困难重重，但可喜的是最终都解决了。最后，在此次课程设计过程中，我还要感谢同学的鼓励和老师给我的支持和建议，让我能顺利的完成本次课程设计。

# 附：代码

**主活动代码：**

public class MainActivity extends BaseActivity {  
  
 //定义是否退出的标志  
 private boolean isExit = false;  
 //定义接收用户发送信息的handler  
 private Handler handler = new Handler() {  
 @Override  
 public void handleMessage(@NonNull Message msg) {  
 super.handleMessage(msg);  
 //标记用户不退出状态  
 isExit = false;  
 }  
 };  
  
 private TextView tv\_video;//视频  
 private TextView tv\_audio;//音乐  
 private ViewPager view\_pager;//视图翻页工具，提供了多页面切换的效果  
 private View view\_indicator;//指示器  
 private int indicatorWidth;//指示器宽度  
  
 /\*\*  
 \* 监听手机物理按键单击事件（退出时需要按两次返回按钮）  
 \* @param keyCode  
 \* @param event  
 \* @return  
 \*/  
 @Override  
 public boolean onKeyDown(int keyCode, KeyEvent event) {  
 if (keyCode == KeyEvent.KEYCODE\_BACK) {  
 //当isExit=false时，提示用户再次按键  
 if (!isExit) {  
 isExit = true;  
 Toast.makeText(getApplicationContext(), "再按一次退出程序", Toast.LENGTH\_SHORT).show();  
 //当用户没有在2秒内再次按返回键时，发送消息标记用户为不退出状态  
 handler.sendEmptyMessageDelayed(0, 2000);  
 } else {  
 //当isExit=true时，退出程序  
 finish();  
 System.exit(0);  
 }  
 }  
 return false;  
 }  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 //解决按home，再次点击图标重新启动的问题 !=0 已经存在任务栈, == 0 不存在Activity实例  
 if ((getIntent().getFlags() & Intent.FLAG\_ACTIVITY\_BROUGHT\_TO\_FRONT) == 0) {  
 //进入闪屏页  
 Intent intent = new Intent(this, SplashActivity.class);  
 startActivity(intent);  
 }  
 super.onCreate(savedInstanceState);  
 }  
  
 //返回一个用于显示界面的布局ID  
 @Override  
 public int getLayoutResID() {  
 return R.layout.activity\_main;  
 }  
  
 //初始化View  
 @Override  
 public void initView() {  
 tv\_video = findView(R.id.tv\_video);  
 tv\_audio = findView(R.id.tv\_audio);  
 view\_indicator = findView(R.id.view\_indicator);  
 view\_pager = findView(R.id.view\_pager);  
 //初始化指示器  
 int screenWidth = Utils.getScreenWidth(this);  
 indicatorWidth = screenWidth / 2;  
 view\_indicator.getLayoutParams().width = indicatorWidth;  
 view\_indicator.requestLayout();//通知这个view去更新它的布局参数  
 }  
  
 //初始化监听器  
 @Override  
 public void initListener() {  
 tv\_video.setOnClickListener(this);  
 tv\_audio.setOnClickListener(this);  
 view\_pager.setOnPageChangeListener(new OnPageChangeListener() {  
  
 // 当页面被选择的时候  
 @Override  
 public void onPageSelected(int position) {  
 changeTitleTextState(position == 0);  
 }  
  
 // 当页面滚动的时候  
 @Override  
 public void onPageScrolled(int position, float positionOffset, int positionOffsetPixels) {  
 //滚动指示线  
 float translationX = indicatorWidth \* position + indicatorWidth \* positionOffset;  
 ViewHelper.setTranslationX(view\_indicator, translationX);  
 }  
  
 // 当页面滚动状态发生改变的时候  
 @Override  
 public void onPageScrollStateChanged(int state) {  
 }  
 });  
 }  
  
 //初始化数据  
 @Override  
 public void initData() {  
 // 默认选中视频  
 changeTitleTextState(true);  
 // 初始化ViewPager  
 ArrayList<Fragment> fragments = new ArrayList<Fragment>();  
 fragments.add(new AudioFragment());//音频  
 fragments.add(new VideoFragment());//视频  
 //设置adapter  
 MainAdapter adapter = new MainAdapter(getSupportFragmentManager(), fragments);  
 view\_pager.setAdapter(adapter);  
 //从音乐界面回来,返回到音乐fragment  
 if (getIntent().getBooleanExtra(Keys.IS\_FROM\_\_AUDIO\_PLAYER, false)) {  
 view\_pager.setCurrentItem(0);  
 }  
 }  
  
 //单击事件  
 @Override  
 public void onClick(View v, int id) {  
 switch (id) {  
 case R.id.tv\_video:  
 view\_pager.setCurrentItem(0);  
 break;  
 case R.id.tv\_audio:  
 view\_pager.setCurrentItem(1);  
 break;  
 default:  
 break;  
 }  
 }  
  
 /\*\*  
 \* 改变标题状态  
 \* @param isSelectVideo 是否选择了视频  
 \*/  
 private void changeTitleTextState(boolean isSelectVideo) {  
 // 改变文本大小  
 scaleTitle(isSelectVideo ? 1.3f : 1.0f, tv\_video);  
 scaleTitle(!isSelectVideo ? 1.3f : 1.0f, tv\_audio);  
 }  
  
 /\*\*  
 \* 缩放标题  
 \* @param scale 缩放的比例  
 \* @param textViw  
 \*/  
 private void scaleTitle(float scale, TextView textViw) {  
 ViewPropertyAnimator.animate(textViw).scaleX(scale).scaleY(scale).setDuration(200);  
 }  
}

**视频播放活动代码：**

public class VideoPlayerActivity extends BaseActivity {

private static final int SHOW\_SYSTEM\_TIME = 0;//显示系统时间

private static final int UPDATE\_PLAYP\_ROGRESS = 1;//更新播放进度

private static final int HIDE\_CTRL\_LAYOUT = 2;//隐藏控制面板

private MyVideoView videoView;//播放界面

OnSeekBarChangeListener mOnVideoSeekBarChangeListener = new OnSeekBarChangeListener() {

@Override

public void onStopTrackingTouch(SeekBar seekBar) {

}

@Override

public void onStartTrackingTouch(SeekBar seekBar) {

}

@Override

public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {

if (fromUser) {

videoView.seekTo(progress);

}

}

};

private VideoItem currentVideo;//当前视频

private TextView tv\_title;//视频名

private ImageView iv\_battary;//电量显示

private TextView tv\_system\_time;//系统时间

private TextView tv\_current\_position;//当前播放位置

private TextView tv\_duration;//总时长

private SeekBar sb\_voice;//音量控制

private SeekBar sb\_video;//进度控制

private Button btn\_pre;//上一个

private Button btn\_next;//下一个

private Button btn\_play;//播放

private Button btn\_fullscreen;//全屏

private BroadcastReceiver batteryChangeReceiver;

private int maxVolume;//最大音量

private AudioManager audioManager;

OnSeekBarChangeListener mOnVoiceSeekBarChangeListener = new OnSeekBarChangeListener() {

// 停止拖动SeekBar

@Override

public void onStopTrackingTouch(SeekBar seekBar) {

}

// 开始拖动SeekBar

@Override

public void onStartTrackingTouch(SeekBar seekBar) {

}

// 进度发生改变的时候

// fromUser 表明是否是由用户出发的

@Override

public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {

if (fromUser) {

setStreamVolume(progress);

}

}

};

private int currentVolume;//

private GestureDetector gestureDetector;//手势检测器

private float currentBrightness;

private ArrayList<VideoItem> videos;

private int currentPosition;

private LinearLayout ll\_top\_ctrl;//顶部控制

private LinearLayout ll\_bottom\_ctrl;//底部控制

private Handler handler = new Handler() {

public void handleMessage(android.os.Message msg) {

switch (msg.what) {

case SHOW\_SYSTEM\_TIME:

showSystemTime();//显示系统时间

break;

case UPDATE\_PLAYP\_ROGRESS:

updatePlayProgress();//更新进度条

break;

case HIDE\_CTRL\_LAYOUT:

showOrHideCtrlLayout();

break;

}

}

;

};

SimpleOnGestureListener mOnGestureListener = new SimpleOnGestureListener() {

//是否是在屏幕左边按下的

private boolean isDownLeft;

@Override

public void onLongPress(MotionEvent e) {

toggleFullScreen();

}

@Override

public boolean onScroll(MotionEvent e1, MotionEvent e2, float distanceX, float distanceY) {

//计算在屏幕y方向滑动的距离

float distanceYY = e1.getY() - e2.getY();

if (isDownLeft) {

changeBrightness(distanceYY);// 如果在屏幕左边按下,则改变屏幕亮度

} else {

changeVolume(distanceYY);// 如果在屏幕右边按下,则改变音量

}

return true;

}

@Override

public boolean onDown(MotionEvent e) {

currentVolume = getStreamVolume();

isDownLeft = e.getX() < Utils.getScreenWidth(VideoPlayerActivity.this) / 2;

// 当前的activity按下时亮度

currentBrightness = getActivityBrightness(VideoPlayerActivity.this);

return super.onDown(e);

}

//双击

@Override

public boolean onDoubleTap(MotionEvent e) {

play();

return true;

}

//单击

@Override

public boolean onSingleTapConfirmed(MotionEvent e) {

showOrHideCtrlLayout();

return true;

}

};

private Uri videoUri;

private ProgressBar pb\_loading;

OnPreparedListener mOnPreparedListener = new OnPreparedListener() {

@Override

public void onPrepared(MediaPlayer mp) {

videoView.start();

updatePlayBtnBg(); // 更新播放按钮背景图片

if (videoUri != null) {

tv\_title.setText(videoUri.getPath());// 第三方应用跳转过来

} else {

tv\_title.setText(currentVideo.getTitle());// 显示视频标题

}

tv\_duration.setText(Utils.formatMillis(videoView.getDuration()));// 视频总时长

sb\_video.setMax(videoView.getDuration());// 设置seekbar最大值

updatePlayProgress();

hideLoading();

mp.setOnBufferingUpdateListener(new OnBufferingUpdateListener() {

@Override

public void onBufferingUpdate(MediaPlayer mp, int percent) {

updateVideoSecondProgress(percent);

}

});

}

};

/\*\*

\* 视屏播放完会回调这个监听器

\*/

OnCompletionListener mOnCompletionListener = new OnCompletionListener() {

@Override

public void onCompletion(MediaPlayer mp) {

videoView.seekTo(0);

tv\_current\_position.setText(Utils.formatMillis(0));

sb\_video.setProgress(0);

updatePlayBtnBg();

next();

}

};

OnInfoListener onInfoListener = new OnInfoListener() {

@Override

public boolean onInfo(MediaPlayer mp, int what, int extra) {

switch (what) {

case MediaPlayer.MEDIA\_INFO\_BUFFERING\_START:

pb\_loading.setVisibility(View.VISIBLE);

return true;

case MediaPlayer.MEDIA\_INFO\_BUFFERING\_END:

pb\_loading.setVisibility(View.GONE);

return true;

}

return false;

}

};

//返回一个用于显示界面的布局ID

@Override

public int getLayoutResID() {

return R.layout.activity\_video\_player;

}

//初始化View

@Override

public void initView() {

videoView = findView(R.id.video\_view);

tv\_title = findView(R.id.tv\_title);

iv\_battary = findView(R.id.iv\_battery);

tv\_system\_time = findView(R.id.tv\_system\_time);

tv\_current\_position = findView(R.id.tv\_current\_position);

tv\_duration = findView(R.id.tv\_duration);

sb\_voice = findView(R.id.sb\_voice);

sb\_video = findView(R.id.sb\_video);

btn\_pre = findView(R.id.btn\_pre);

btn\_next = findView(R.id.btn\_next);

btn\_play = findView(R.id.btn\_play);

btn\_fullscreen = findView(R.id.btn\_fullscreen);

pb\_loading = findView(R.id.pb\_loading);

showSystemTime();

initCtrlLayout();

}

//初始化控制面板

private void initCtrlLayout() {

ll\_top\_ctrl = findView(R.id.ll\_top\_ctrl);

ll\_bottom\_ctrl = findView(R.id.ll\_bottom\_ctrl);

// 顶部控制栏的隐藏: Y方向移动控件的高度的负数

ll\_top\_ctrl.measure(0, 0);

ViewHelper.setTranslationY(ll\_top\_ctrl, -ll\_top\_ctrl.getMeasuredHeight());

// 底部控制栏的隐藏,Y方向移动控件的高度

ll\_bottom\_ctrl.measure(0, 0);

ViewHelper.setTranslationY(ll\_bottom\_ctrl, ll\_bottom\_ctrl.getMeasuredHeight());

}

//显示系统时间

private void showSystemTime() {

tv\_system\_time.setText(DateFormat.format("kk:mm:ss", System.currentTimeMillis()));

handler.sendEmptyMessageDelayed(SHOW\_SYSTEM\_TIME, 1000);

}

@Override

protected void onDestroy() {

handler.removeCallbacksAndMessages(null);

unregisterReceiver(batteryChangeReceiver);

super.onDestroy();

}

//初始化监听器

@Override

public void initListener() {

videoView.setOnPreparedListener(mOnPreparedListener);

videoView.setOnCompletionListener(mOnCompletionListener);

videoView.setOnInfoListener(onInfoListener);

sb\_voice.setOnSeekBarChangeListener(mOnVoiceSeekBarChangeListener);

sb\_video.setOnSeekBarChangeListener(mOnVideoSeekBarChangeListener);

gestureDetector = new GestureDetector(this, mOnGestureListener);

}

@SuppressWarnings("unchecked")

@Override

public void initData() {

videoUri = getIntent().getData();

if (videoUri != null) {

// 从第三方跳转过来

pb\_loading.setVisibility(View.VISIBLE);

videoView.setVideoURI(videoUri);

btn\_next.setEnabled(false);

btn\_pre.setEnabled(false);

} else {

// 从视频列表跳转过来的

videos = (ArrayList<VideoItem>) getIntent().getSerializableExtra(Keys.ITEMS);

currentPosition = getIntent().getIntExtra(Keys.CURRENT\_POSITION\_IN\_LIST, -1);

openVideo();

}

registerBatteryChangeReceiver();

initVoice();

}

/\*\*

\* 打开一个视屏

\*/

private void openVideo() {

if (videos == null || videos.isEmpty() || currentPosition == -1) {

return;

}

btn\_pre.setEnabled(currentPosition != 0);

btn\_next.setEnabled(currentPosition != videos.size() - 1);

currentVideo = videos.get(currentPosition);

String path = currentVideo.getData();

pb\_loading.setVisibility(View.VISIBLE);

videoView.setVideoPath(path);

}

/\*\*

\* 初始化音量

\*/

private void initVoice() {

audioManager = (AudioManager) getSystemService(Context.AUDIO\_SERVICE);

maxVolume = audioManager.getStreamMaxVolume(AudioManager.STREAM\_MUSIC);

int currentVolume = getStreamVolume();

sb\_voice.setMax(maxVolume);

sb\_voice.setProgress(currentVolume);

}

/\*\*

\* 获取当前音量

\*/

private int getStreamVolume() {

return audioManager.getStreamVolume(AudioManager.STREAM\_MUSIC);

}

/\*\*

\* 设置音量

\* @param value 音量值

\*/

private void setStreamVolume(int value) {

int flags = 0;// 1-显示系统的音量调节面板,0-不显示系统的音量调节面板

audioManager.setStreamVolume(AudioManager.STREAM\_MUSIC, value, flags);

}

/\*\*

\* 注册电量改变广播接收者

\*/

private void registerBatteryChangeReceiver() {

batteryChangeReceiver = new BroadcastReceiver() {

@Override

public void onReceive(Context context, Intent intent) {

// 获取电量等级

int level = intent.getIntExtra(BatteryManager.EXTRA\_LEVEL, -1);

updateBatteryBg(level);

}

};

IntentFilter filter = new IntentFilter(Intent.ACTION\_BATTERY\_CHANGED);

registerReceiver(batteryChangeReceiver, filter);

}

/\*\*

\* 更新电量显示图片

\* @param level 当前的电量级别

\*/

protected void updateBatteryBg(int level) {

Logger.i(this, "Batterylevel:" + level);

int resid = R.drawable.ic\_battery\_0;

if (level == 0) {

resid = R.drawable.ic\_battery\_0;

} else if (level >= 100) {

resid = R.drawable.ic\_battery\_100;

} else if (level >= 80) {

resid = R.drawable.ic\_battery\_80;

} else if (level >= 60) {

resid = R.drawable.ic\_battery\_60;

} else if (level >= 40) {

resid = R.drawable.ic\_battery\_40;

} else if (level >= 20) {

resid = R.drawable.ic\_battery\_20;

} else if (level >= 10) {

resid = R.drawable.ic\_battery\_10;

}

iv\_battary.setBackgroundResource(resid);

}

@Override

public void onClick(View v, int id) {

switch (id) {

case R.id.btn\_voice: // 静音按钮

mute();

break;

case R.id.btn\_exit: // 退出按钮

finish();

break;

case R.id.btn\_pre: // 上一首按钮

pre();

break;

case R.id.btn\_play: // 播放按钮

play();

break;

case R.id.btn\_next: // 下一首按钮

next();

break;

case R.id.btn\_fullscreen:// 全屏按钮

toggleFullScreen();

break;

default:

break;

}

}

/\*\*

\* 在全屏和默认大小之间切换

\*/

private void toggleFullScreen() {

videoView.switchFullScreen();

updateFullScreenBtnBg();

}

/\*\*

\* 更新全屏按钮的背景

\*/

private void updateFullScreenBtnBg() {

int resid;

if (videoView.isFullScreen()) {

resid = R.drawable.ic\_btn\_video\_small;// 显示恢复默认大小背景

} else {

resid = R.drawable.ic\_btn\_video\_big;// 显示全屏按钮

}

btn\_fullscreen.setBackgroundResource(resid);

}

/\*\*

\* 播放或暂停

\*/

private void play() {

if (videoView.isPlaying()) {

videoView.pause();// 如果当前是正在播放的,则暂停

} else {

videoView.start(); // 如果当前是暂停的,则播放

}

updatePlayBtnBg();

}

private void updatePlayBtnBg() {

int resid;

if (videoView.isPlaying()) {

resid = R.drawable.ic\_btn\_video\_pause;// 如果当前是正在播放的,则显示暂停按钮

} else {

resid = R.drawable.ic\_btn\_video\_play;// 如果当前是暂停的,则显示播放按钮

}

btn\_play.setBackgroundResource(resid);

}

/\*\*

\* 上一首

\*/

private void pre() {

if (currentPosition != 0) {

currentPosition--;

openVideo();

}

}

/\*\*

\* 下一首

\*/

private void next() {

if (videos == null) {

finish();

}

if (currentPosition != videos.size() - 1) {

currentPosition++;

openVideo();

}

}

/\*\*

\* 静音或者恢复原来的音量

\*/

private void mute() {

if (getStreamVolume() > 0) {

// 如果当前音量大于0,则保存一下这个音量然后设置为0

currentVolume = getStreamVolume();

setStreamVolume(0);

sb\_voice.setProgress(0);

} else {

// 如果当前音量为0,则恢复原来保存的音量

setStreamVolume(currentVolume);

sb\_voice.setProgress(currentVolume);

}

}

/\*\*

\* 更新视屏缓冲进度

\*

\* @param percent 缓冲进度百分比

\*/

private void updateVideoSecondProgress(int percent) {

float percentFloat = percent / 100.0f;

int secondaryProgress = (int) (videoView.getDuration() \* percentFloat);

sb\_video.setSecondaryProgress(secondaryProgress);

}

/\*\*

\* 使用渐变的方式慢慢隐藏Loading界面

\*/

protected void hideLoading() {

pb\_loading.setVisibility(View.GONE);

}

/\*\*

\* 跟新播放进度

\*/

protected void updatePlayProgress() {

// 显示当前播放位置

tv\_current\_position.setText(Utils.formatMillis(videoView.getCurrentPosition()));

// 设置当前播放进度

sb\_video.setProgress(videoView.getCurrentPosition());

handler.sendEmptyMessageDelayed(UPDATE\_PLAYP\_ROGRESS, 300);

}

public boolean onTouchEvent(MotionEvent event) {

// 把触摸事件传递给手势监听器

boolean result = gestureDetector.onTouchEvent(event);

return result;

}

@Override

public boolean dispatchTouchEvent(MotionEvent ev) {

switch (ev.getAction()) {

case MotionEvent.ACTION\_DOWN:

cancelHideCtrlLayoutMessage();

break;

case MotionEvent.ACTION\_UP:

sendHideCtrlLayoutMessage();

break;

}

return super.dispatchTouchEvent(ev);

}

/\*\*

\* 根据滑动屏幕的距离改变音量

\* @param distanceY Y方向移动屏幕的距离

\*/

private void changeVolume(float distanceY) {

// 4.计算滑动的距离等于多少对应的音量

// a.音量最大值与屏幕高的比例

float scale = ((float) maxVolume) / Utils.getScreenHeight(VideoPlayerActivity.this);

// b.计算滑动的距离等于多少对应的音量值

int moveVolume = (int) (distanceY \* scale \* 2);

// 5.在原来的音量的基础上加上计算出来的对应音量值

int resultVolume = currentVolume + moveVolume;

if (resultVolume > maxVolume) {

resultVolume = maxVolume;

} else if (resultVolume < 0) {

resultVolume = 0;

}

Logger.i(this, "resultVolume:" + resultVolume + "\*\*\*\*\*\*" + distanceY);

// 6.使用这个音量值

setStreamVolume(resultVolume);

sb\_voice.setProgress(resultVolume);

}

/\*\*

\* 改变屏幕亮度

\*/

protected void changeBrightness(float distanceY) {

// 4.计算滑动的距离等于多少对应的亮度

// a.亮度最大值与屏幕高的比例

float scale = (1.0f) / Utils.getScreenHeight(VideoPlayerActivity.this);

// b.计算滑动的距离等于多少对应的亮度值

float moveBrightness = distanceY \* scale \* 2;

// 5.在原来的亮度的基础上加上计算出来的对应亮度值

float resultBrightness = currentBrightness + moveBrightness;

if (resultBrightness > 1.0f) {

resultBrightness = 1.0f;

} else if (resultBrightness < 0) {

resultBrightness = 0;

}

Logger.i(this, "resultBrightness:" + resultBrightness + "\*\*\*\*\*\*" + distanceY);

// 6.使用这个亮度值

setActivityBrightness(resultBrightness, this);

}

/\*\*

\* 显示或隐藏控制面板

\*/

protected void showOrHideCtrlLayout() {

if (ViewHelper.getTranslationY(ll\_top\_ctrl) == 0) {

// 如果控制面板原来是显示的,则隐藏

// 顶部控制栏的隐藏: Y方向移动控件的高度的负数

ViewPropertyAnimator.animate(ll\_top\_ctrl).translationY(-ll\_top\_ctrl.getHeight());

// 底部控制栏的隐藏,Y方向移动控件的高度

ViewPropertyAnimator.animate(ll\_bottom\_ctrl).translationY(ll\_bottom\_ctrl.getHeight());

} else {

// 如果控制面板原来是隐藏的,则显示

ViewPropertyAnimator.animate(ll\_top\_ctrl).translationY(0);

ViewPropertyAnimator.animate(ll\_bottom\_ctrl).translationY(0);

sendHideCtrlLayoutMessage();

}

}

/\*\*

\* 发送隐藏控制面板的消息,5秒后执行

\*/

private void sendHideCtrlLayoutMessage() {

cancelHideCtrlLayoutMessage();

handler.sendEmptyMessageDelayed(HIDE\_CTRL\_LAYOUT, 3000);

}

/\*\*

\* 取消隐藏控制面板消息

\*/

private void cancelHideCtrlLayoutMessage() {

handler.removeMessages(HIDE\_CTRL\_LAYOUT);

}

/\*\*

\* 获取当前activity的屏幕亮度

\* @param activity 当前的activity对象

\* @return 亮度值范围为0-1.0f，如果为-1.0，则亮度与全局同步,没设置亮度默认返回-1。

\*/

public float getActivityBrightness(Activity activity) {

Window localWindow = activity.getWindow();

WindowManager.LayoutParams params = localWindow.getAttributes();

if (params.screenBrightness < 0) {

return getScreenBrightness() / 255.0f;

}

return params.screenBrightness;

}

/\*\*

\* 设置当前activity的屏幕亮度

\* @param paramFloat 0-1.0f,当传入参数-1.0f，当前的activity就又会重新使用全局的亮度属性

\* @param activity 需要调整亮度的activity

\*/

public void setActivityBrightness(float paramFloat, Activity activity) {

Window localWindow = activity.getWindow();

WindowManager.LayoutParams params = localWindow.getAttributes();

params.screenBrightness = paramFloat;

localWindow.setAttributes(params);

}

/\*\*

\* 获得当前屏幕亮度值

\* @return 0--255

\*/

public int getScreenBrightness() {

int screenBrightness = -1;

try {

screenBrightness = Settings.System.getInt(getContentResolver(), Settings.System.SCREEN\_BRIGHTNESS);

} catch (Settings.SettingNotFoundException e) {

e.printStackTrace();

}

return screenBrightness;

}

}