多媒体应用项目方案

【2019年2月】

内部资料保密勿泄漏

## 版本记录

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **日期** | **版本号** | **修订说明** | **修订人** | **审核人** | **批准人** |
| 2019.02.12 | V1.0 |  | 唐石元 |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**目录**

[一、 前言 1](#_Toc338346695)

[二、 项目背景及需求分析 2](#_Toc338346696)

[2.1 项目背景 2](#_Toc338346697)

[2.2 项目需求 2](#_Toc338346698)

[2.3 其它需求 2](#_Toc338346699)

[三、 项目设计概要 8](#_Toc338346700)

# 前言概述

本方案的宗旨在于指导项目设计人员，开发人员更好的进行多媒体项目的设计开发及项目的后续维护和改进。

# 项目背景及需求分析

## 项目背景

本项目基于车载芯片ATC AC8227 Android6.0开发，兼容Android8.0。基于安卓车机平台提供本地多媒体文件的播放、浏览功能。按具体功能划分为媒体扫描模块、视频播放应用、音乐播放应用、图片播放应用四个部分。

## 项目需求

### 媒体扫描模块

1．以服务接口的方式提供本地存储设备视频、音频、图片的快速扫描搜索功能。可实现不同存储设备的指定切换（如下图）。



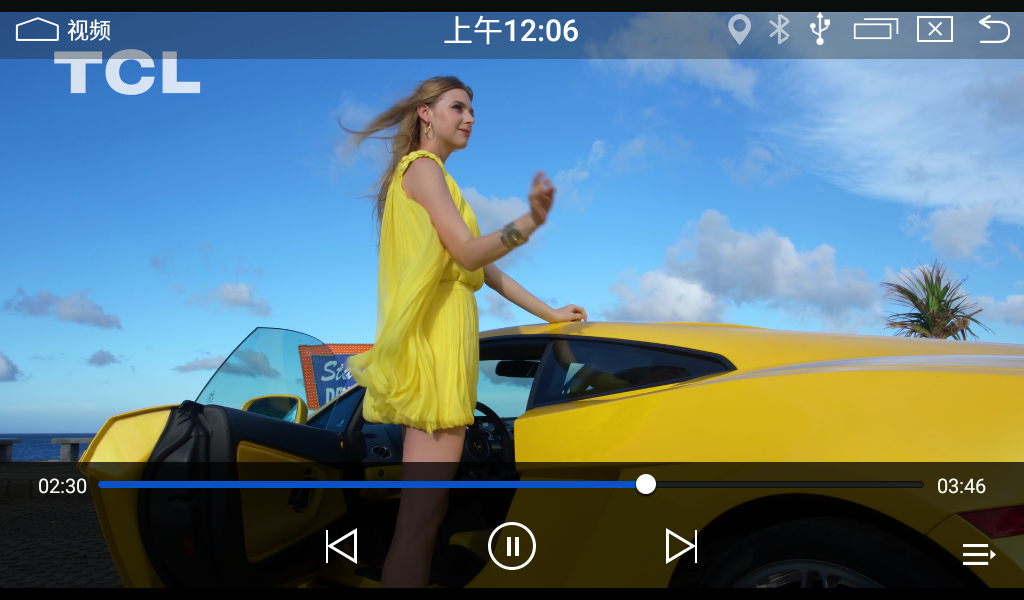
2．媒体文件支持格式如下：

|  |  |  |
| --- | --- | --- |
| **多媒体应用支持文件格式** | | |
| **视频文件格式** | **音频文件格式** | **图片文件格式** |
| **mp4** | **aac** | **png** |
| **mkv** | **amr** | **bmp** |
| **mov** | **ape** | **gif** |
| **ts** | **flac** | **ico** |
| **avi** | **m4a** | **jpg** |
| **3gp** | **mka** |  |
| **3gpp** | **mp3** |  |
| **3g2** | **ogg** |  |
| **flv** | **wav** |  |
| **mpeg** |  |  |
| **mpg** |  |  |
| **rm** |  |  |
| **tp** |  |  |
| **vop** |  |  |
| **wmv** |  |  |

3．解析音频文件的ID3信息。

### 视频播放应用

功能：实现本地视频文件的播放功能(如下图)。





1. 提供进度条控制显示播放时间，可暂停播放，可跳转上下集。
2. 可按存储器列表，所有播放列表，文件夹列表选择播放视频。

### 音频播放应用

功能：实现本地音频文件的播放功能(如下图)。

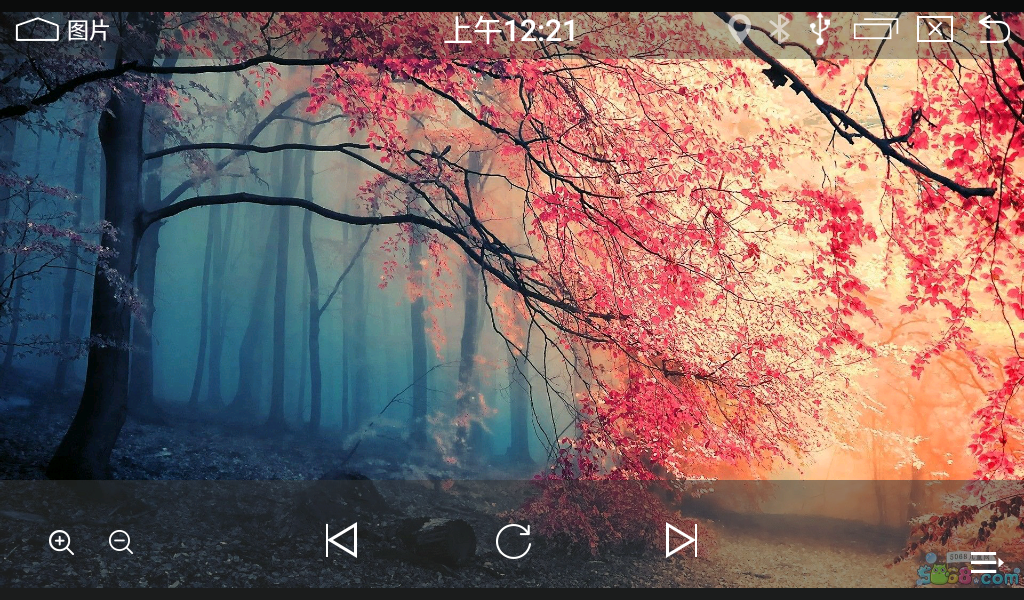




1. 播放当前歌曲时，显示id3信息中的图片，歌手，专辑等信息。提供进度条控制显示播放时间，可暂停播放，可跳转上下曲。
2. 可按存储器，单曲，歌手，专辑，文件夹选择播放歌曲。
3. 提供顺序循环，单曲循环，随机播放，歌手循环四种连续播放模式。

### 图片播放应用

功能：实现本地图片文件的播放功能(如下图)。





1. 提供放大缩小、图片旋转，可跳转上下张图片。
2. 可按存储器列表，所有图片列表，文件夹列表选择浏览图片。
3. 提供触摸控制缩放功能。

## 其它需求(车机特有)

1. **视频播放应用需要提供同步监听检测刹车状态，刹车状态下方能播放视频。**
2. **音乐播放应用在设置选项中可以设置插入U盘是否立即打开音乐播放功能。（现在的设计为音乐播放应用不检测U盘是否插入，由中控控制启动音乐播放应用，同媒体扫描模块切换插入U盘，从而实现播放刚插入U盘歌曲的功能）**
3. **导航报点时可根据设置决定音视频播放应用是否需要暂停播放或者降低音量混音播放。**
4. **接打电话、倒车时需要暂停播放，挂断电话、倒车结束恢复播放。**
5. **音乐播放应用需上传正在播放歌曲的ID3信息给中控。**
6. **音乐播放应用接受Intent传入参数启动，用播放语音控制请求打开的歌曲。**

# 项目设计概要

## 模块功能层次划分



备注说明：其中多媒体常用类库封装基于对开发视频播放、音频播放、图片播放比较通用可以重复使用的一些接口、基类等进行了封装，并实现了与媒体扫描服务交互，便于以后更换UI进行二次开发以及再次开发类似应用时重复使用。

媒体扫描服务定义规范了相应接口，便于今后扩展优化，可修改对应实现接口的逻辑算法。

## 媒体扫描模块

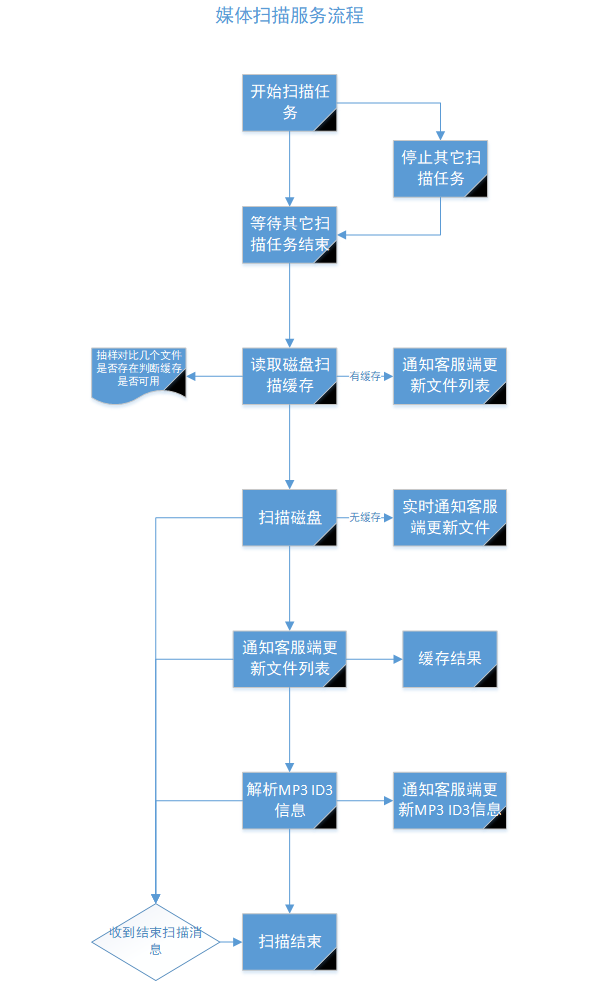
1. 接口定义。
2. **接收扫描结果需要的接口定义**：

*/\*\*  
 \* 通知扫描服务返回的音乐列表  
 \*/* **void** notifyMusic(**inout** List<Music> list);  
  
 */\*\*  
 \* 通知扫描服务返回的视频列表  
 \*/* **void** notifyVideo(**inout** List<Video> list);  
  
 */\*\*  
 \* 通知扫描服务返回的图片列表  
 \*/* **void** notifyImage(**inout** List<Image> list);  
  
 */\*\*  
 \* 通知扫描服务返回的已有Id3信息的音乐列表  
 \*/* **void** notifyID3Music(**inout** List<Music> list);  
  
 */\*\*  
 \* 通知扫描服务将对指定路径开始扫描  
 \*/* **void** notifyPath(String path);  
  
 */\*\*  
 \* 通知扫描服务已完成对指定路径的扫描  
 \*/* **void** notifyFinish(String path);

**②与扫描模块进行交互需要的接口定义**：

*/\*\*  
 \* 通知扫描服务扫描指定路径。  
 \*/* **void** scanDisk(String disk);  
  
 */\*\*  
 \* 注册接收扫描获取的信息通知  
 \*/* **void** registerNotify(Notify notify);  
  
 */\*\*  
 \* 注销接收扫描获取的信息通知  
 \*/* **void** unregisterNotify(Notify notify);

1. 扫描模块工作流程图。



## 视频播放应用

1．视频解码实现：视频播放基于第三方开源播放框架IjkPlayer修改，在原有支持硬解格式(h264,hevc,mpeg2,mpeg4)的基础上添加了对h263,wmv3硬解支持，因平台本身对音频格式的支持有限，对音频的解码统一使用ffmpeg软解。

添加相应硬解支持修改代码如下：

//@FlyZebra ATC8227L ADD

case AV\_CODEC\_ID\_H263:

ALOGE("AV\_CODEC\_ID\_H263");

if (!ffp->mediacodec\_all\_videos) {

ALOGE("%s: MediaCodec/AV\_CODEC\_ID\_H263 is disabled. codec\_id:%d \n", \_\_func\_\_, opaque->codecpar->codec\_id);

goto fail;

}

strcpy(opaque->mcc.mime\_type, SDL\_AMIME\_VIDEO\_H263);

opaque->mcc.profile = opaque->codecpar->profile;

opaque->mcc.level = opaque->codecpar->level;

break;

//@FlyZebra ATC8227L ADD

case AV\_CODEC\_ID\_WMV3:

// goto fail;

ALOGE("AV\_CODEC\_ID\_WMV3");

if (!ffp->mediacodec\_all\_videos) {

ALOGE("%s: MediaCodec/AV\_CODEC\_ID\_WMV3 is disabled. codec\_id:%d \n", \_\_func\_\_, opaque->codecpar->codec\_id);

goto fail;

}

strcpy(opaque->mcc.mime\_type, "video/x-ms-wmv");

opaque->mcc.profile = opaque->codecpar->profile;

opaque->mcc.level = opaque->codecpar->level;

break;

2．UI界面设计实现的几点细节：

①文件列表使用RecyclerView实现，关键实现代码如下：

@Override  
**public void** onAttachedToRecyclerView(**final** RecyclerView recyclerView) {  
 GridLayoutManager gridLayoutManager = (GridLayoutManager) recyclerView.getLayoutManager();  
 gridLayoutManager.setSpanSizeLookup(**new** GridLayoutManager.SpanSizeLookup() {  
 @Override  
 **public int** getSpanSize(**int** position) {  
 **switch** (**mList**.get(position).**type**) {  
 **case** 1:  
 **return mColumnNum**;  
 **case** 2:  
 **return** 1;  
 **case** 3:  
 **return mColumnNum**;  
 }  
 **return mColumnNum**;  
 }  
 });  
}

这样可以让不同的数据类型占用不同的列数，从而可以同时显示文件项和图片项内容。

② 全屏需要同步SystemUI一起显示/隐藏菜单，关键代码如下：

getWindow().getDecorView().setOnSystemUiVisibilityChangeListener(**new** View.OnSystemUiVisibilityChangeListener() {  
 @Override  
 **public void** onSystemUiVisibilityChange(**int** visibility) {  
 **if** (visibility == 0) {  
 showControlView(**true**);  
 }  
 }  
});

## 音频播放应用

1. 音频解码使用Android原生MediaPlayer，只支持播放硬件支持的音乐格式。
2. UI界面设计实现的几点细节：
   1. 歌手、专辑、文件夹列表使用ExpandableListView实现。

3．ID3解析使用第三方库,Github地址: https://github.com/mpatric/mp3agic

## 图片播放应用.

1. 图片显示使用开源PhotoView，解决某些情况下旋转更新不及时的问题，修改了其源码文件PhotoViewAttacher.java

**private void** setImageViewMatrix(Matrix matrix) {  
 FlyLog.*d*(**"setImageViewMatrix matrix="**+matrix);  
 **mImageView**.setImageMatrix(matrix);  
 **mImageView**.requestLayout();  
  
 // Call MatrixChangedListener if needed  
 **if** (**mMatrixChangeListener** != **null**) {  
 RectF displayRect = getDisplayRect(matrix);  
 **if** (displayRect != **null**) {  
 **mMatrixChangeListener**.onMatrixChanged(displayRect);  
 }  
 }  
}

添加行**mImageView**.requestLayout();旋转图片时请求重绘。

1. UI界面设计实现的几点细节：
   1. 文件列表与全屏同以上视频播放应用部分中的实现。
   2. 图片主预览画面使用ViewPager实现，为解决Viewpager翻页浏览的内存泄漏问题，在PagerAdapater中有以下代码：

**private class** MyPageAdapter **extends** PagerAdapter {  
 **private** HashSet<PhotoView> **viewSet** = **new** HashSet<>();  
  
 @Override  
 **public void** destroyItem(ViewGroup container, **int** position, Object object) {  
 ……  
 **viewSet**.add(photoView);  
 container.removeView(photoView);  
 }  
  
 @Override  
 **public** Object instantiateItem(ViewGroup container, **int** position) {  
 FlyLog.*d*(**"set size=%d"**, **viewSet**.size());  
 PhotoView photoView = **null**;  
 Iterator it = **viewSet**.iterator();  
 **if** (it.hasNext()) {  
 photoView = (PhotoView) it.next();  
 **viewSet**.remove(photoView);  
 } **else** {  
 photoView = **new** PhotoView(PhotoActivity\_AP1.**this**);  
 }  
 }  
  
}

**使用viewSet集合重复利用PhotoView，防制生成过多的PhotoView对像。**