

立即体验

学院 (<http://edu.csdn.net?ref=toolbar>)

更多 ▼




登录 (<http://passport.csdn.net/account/login?ref=toolbar>) 注册 (<http://passport.csdn.net/account/mobileregister?ref=toolbar&action=mobileRegister>)

5
rc

原创 2016年12月08日 17:09:30

an

④

 4775

+关注

码云

原创	粉丝	喜欢	2 (https://git /lantingsh
18	11	0	

更多文章 (<http://blog.csdn.net/lantingshuxu>)

【java】java OSGi初体验 (<http://blog.csdn.net/LanTingShuXu/article/details/78202397>)

【Linux】centOS配置JavaWeb环境--
1-->jdk及MySQL配置 (<http://blog.csdn.net/LanTingShuXu/article/details/72801975>)

【数据库】MySQL中删除主键 (<http://blog.csdn.net/LanTingShuXu/article/details/70215063>)

JDK9的新特性
2015/5/22 09:56 (http://edu.csdn.net/course)

jdk9新特性
detail/6134?utm_source=blog9)
(http://edu.csdn.net
course/detail
/6134?utm_source=blog9

SDCC 2017 (http://edu.csdn.net)

http://edu.csdn.net/multi/course_source=blog9/series_detail/73?utm_source=blog9

最近做的项目是和语音实时采集并发送，对方实时接收并播放相关，下面记录下实现的核心代码。

很多android开发者应该知道android有个MediaRecorder对象和MediaPlayer对象，用于录制和播放音频。

这个弊端在于他们不能实时采集并发送出去，所以，我们只能使用AudioRecord和AudioTrack来实现。

记得申明权限：

```
1 <uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS" />
2 <uses-permission android:name="android.permission.RECORD_AUDIO" >
```

1、先申明相关录制配置参数

```
1 private AudioRecord audioRecord;// 录音对象
2 private int frequency = 8000;// 采样率 8000
3 private int channelInConfig = AudioFormat.CHANNEL_CONFIGURATION_MONO;// 定义采样通道
4 private int audioEncoding = AudioFormat.ENCODING_PCM_16BIT;// 定义音频编码（16位）
5 private byte[] buffer = null;// 录制的缓冲数组
```

2、在开始录制前，我们需要初始化AudioRecord类。

```
1 // 根据定义好的几个配置，来获取合适的缓冲大小
2 // int bufferSize = 800;
3 int bufferSize = AudioRecord.getMinBufferSize(frequency,
4         channelInConfig, audioEncoding);
5 // 实例化AudioRecord
6 audioRecord = new AudioRecord(MediaRecorder.AudioSource.MIC,
7         frequency, channelInConfig, audioEncoding, bufferSize);
8 // 定义缓冲数组
9 buffer = new byte[bufferSize];
```

3、准备开始录制，使用循环不断读取数据。

```
1 audioRecord.startRecording();// 开始录制
2 isRecording = true;// 设置录制标记为true
3
4 // 开始录制
5 while (isRecording) {
6 // 录制的内容放置到了buffer中，result代表存储长度
7 int result = audioRecord.read(buffer, 0, buffer.length);
8 /*.....result为buffer中录制数据的长度(貌似基本上都是640)。
9 剩下就是处理buffer了，是发送出去还是直接播放，这个随便你。*/
10 }
11
12 //录制循环结束后，记得关闭录制！！
13 if (audioRecord != null) {
14     audioRecord.stop();
15 }
```

二、AudioTrack代码实现介绍如下：

1、声明播放相关配置。

```
1 private AudioTrack track = null;// 录音文件播放对象
2 private int frequency = 8000;// 采样率 8000
3 private int channelInConfig = AudioFormat.CHANNEL_CONFIGURATION_MONO;// 定义采样通道
4 private int audioEncoding = AudioFormat.ENCODING_PCM_16BIT;// 定义音频编码（16位）
5 private int bufferSize = -1;// 播放缓冲大小
```

2、初始化AudioTrack对象（初始化一次，该对象可重复使用）

```
1 // 获取缓冲 大小
2 bufferSize = AudioTrack.getMinBufferSize(frequency, channelInConfig,
3     audioEncoding);
4 // 实例AudioTrack
5 track = new AudioTrack(AudioManager.STREAM_MUSIC, frequency,
6     channelInConfig, audioEncoding, bufferSize,
7     AudioTrack.MODE_STREAM);
```

3、使用AudioTrack播放语音数据。

```
1 //将语音数据写入即可。
2 track.write(dataArray, buffer, len);
```

他的热门文章

【android开发】实现语音数据实时采集/播放 (<http://blog.csdn.net/lantingshuxu/article/details/53520316>)

4760

【常用技能】使用eclipse自带git插件托管项目到“码云” (<http://blog.csdn.net/lantingshuxu/article/details/53563433>)

3917

【android开发】桌面小挂件(APP Widgets) (<http://blog.csdn.net/lantingshuxu/article/details/53454709>)

1403

【android开发】手势滑动关闭Activity(随手指消失)的辅助类的实现 (<http://blog.csdn.net/lantingshuxu/article/details/53423250>)

1394

【android开发】解决输入法与表情面板切换时的界面抖动问题 (<http://blog.csdn.net/lantingshuxu/article/details/53376070>)

978

相关推荐

【android开发】实现语音数据实时采集/播放 (http://blog.csdn.net/huang_rong12/article/details/53534441)

研究Android即时聊天、实时语音通话、实时对讲机等必备知识Audio (<http://blog.csdn.net/d1020965026/article/details/25118153>)

Android实时语音聊天 (http://blog.csdn.net/lv_ws/article/details/41863149)

Android 多媒体之实现语音聊天界面 (<http://blog.csdn.net/lovoo/article/details/51661077>)

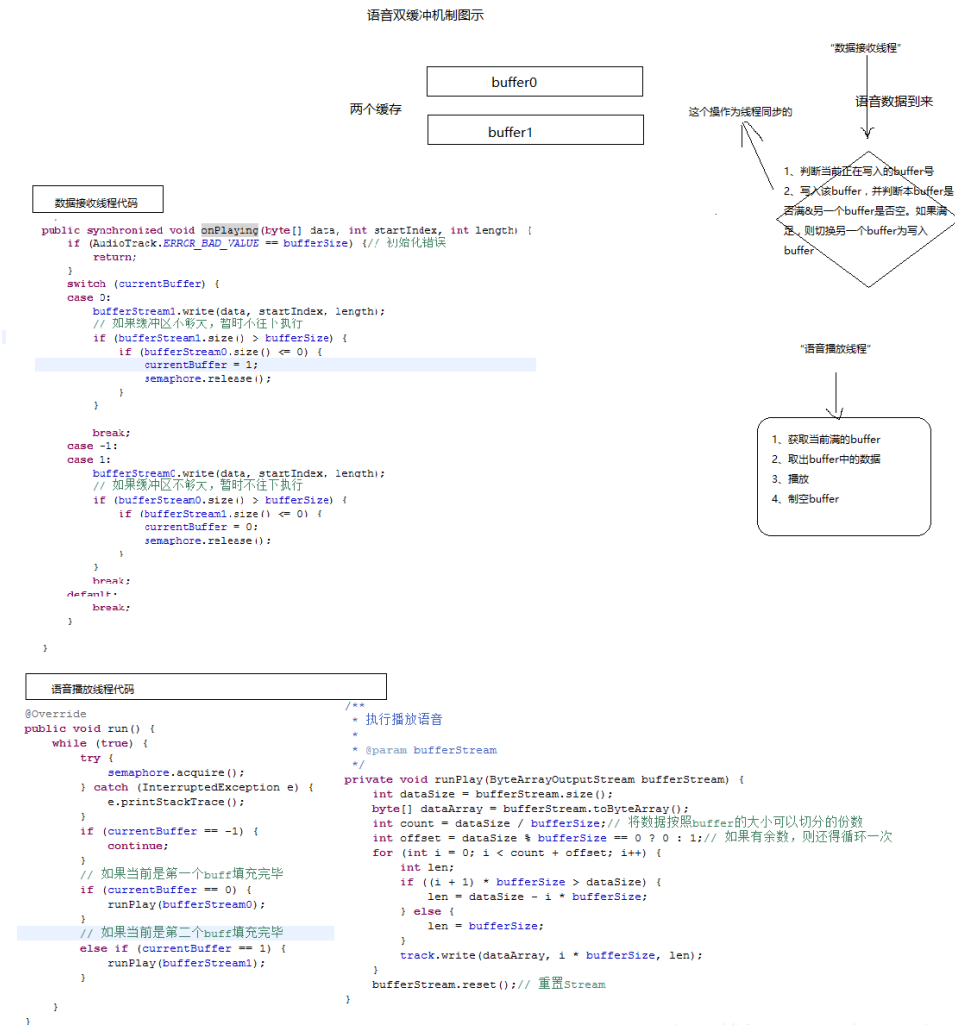
问题一：

由于目前的项目是实时采集，实时发送，所以需要考虑到包的大小，经测试，我们使用160个byte作为一个包传递可以做到比较好的播放效果（也就是将一份buffer拆分成四个发送）。处理代码如下：

```
1 // 将数据通过监听接口回调出去
2 if (audioRecordingCallback != null) {
3     int offset = result % MAX_DATA_LENGTH > 0 ? 1 : 0;
4     // 将一个buffer拆分成几份小数据包 MAX_DATA_LENGTH 为包的最大byte数
5     for (int i = 0; i < result / MAX_DATA_LENGTH + offset; i++) {
6         int length = MAX_DATA_LENGTH;
7         if ((i + 1) * MAX_DATA_LENGTH > result) {
8             length = result - i * MAX_DATA_LENGTH;
9         }
10        // 写到回调接口
11        audioRecordingCallback.onRecording(buffer, i
12        * MAX_DATA_LENGTH, length);
13    }
14 }
```

问题二：

有时候传输的过来播放声音会一卡一卡的，为了解决这样的问题，暂时使用了语音双缓冲机制来解决，问题优化很明显。代码和示意图如下：



<http://blog.csdn.net/LanTingShuXu>

【有朋友说要源码，那我就贴下】

【声音采集的源码】



5



```
1  /**
2   * 实时音频录制处理类<br/>
3   * 记得申明系统权限：MODIFY_AUDIO_SETTINGS、RECORD_AUDIO<br/>
4   * 使用实例代码：<br/>
5   *
6   * <pre>
7   * AudioRecorderHandler = new AudioRecorderHandler(this);
8   * audioRecorderHandler.startRecord(new AudioRecordingCallback() {
9   *     @Override
10    * public void onStopRecord(String savedPath) {
11    *
12    *
13    *
14    *     @Override
15    * public void onRecording(byte[] data, int startIndex, int length) {
16    *     // TODO 录制监听。处理data即可。立即播放或发送出去，随你。
17    * }
18    * });
19    * </pre>
20    *
21    * @author 李长军
22    *
23    */
24    @SuppressWarnings("deprecation")
25    public class AudioRecorderHandler {
26
27        /**
28         * 录音数据单次回调数组最大为多少
29         */
30        private static int MAX_DATA_LENGTH = 160;
31
32        private AudioRecord audioRecord;// 录音对象
33        private boolean isRecording = false;// 标记是否正在录音中
34        private int frequency = 8000;// 采样率 8000
35        private int channelInConfig = AudioFormat.CHANNEL_CONFIGURATION_MONO;// 定义采样通道（过时，但是使用其他的
36        private int audioEncoding = AudioFormat.ENCODING_PCM_16BIT;// 定义音频编码（16位）
37        private byte[] buffer = null;// 录制的缓冲数组
38        private File lastCacheFile = null;// 记录上次录制的文件名
39
40        public CacheCleanManager cacheManager;
41
42        public AudioRecorderHandler(Context context) {
43            if (context == null) {
44                throw new RuntimeException("Context could not be null!");
45            }
46        }
47
48        /**
49         * 开始录制音频
50         *
51         * @param callBackListener
52         *      录制过程中的回调函数
53         */
54        public void startRecord(AudioRecordingCallback audioRecordingCallback) {
55            RecordTask task = new RecordTask(audioRecordingCallback);
56            task.execute();// 开始执行
```

```
57     }
58
59     /**
60     * 停止录制
61     */
62     public void stopRecord() {
63         isRecording = false;
64
65         5
66         /**
67         * 删除上次录制的文件（一般是用户取消发送导致删除上次录制的内容）
68         */
69         @return true表示删除成功，false表示删除失败，一般是没有上次录制的文件，或者文件已经被删除了
70         */
71         public boolean deleteLastRecordFile() {
72             boolean success = false;
73             if (lastCacheFile != null && lastCacheFile.exists()) {
74                 success = lastCacheFile.delete();
75             }
76             return success;
77         }
78
79         /**
80         * 录制音频的任务类
81         *
82         * @author 李长军
83         *
84         */
85         private class RecordTask extends AsyncTask<String, Integer, String> {
86
87             private AudioRecordingCallback audioRecordingCallback = null;
88
89             public RecordTask(AudioRecordingCallback audioRecordingCallback) {
90                 this.audioRecordingCallback = audioRecordingCallback;
91             }
92
93             @Override
94             protected void onPreExecute() {
95                 // 根据定义好的几个配置，来获取合适的缓冲大小
96                 // int bufferSize = 800;
97                 int bufferSize = AudioRecord.getMinBufferSize(frequency,
98                     channelInConfig, audioEncoding);
99                 // 实例化AudioRecord
100                 audioRecord = new AudioRecord(MediaRecorder.AudioSource.MIC,
101                     frequency, channelInConfig, audioEncoding, bufferSize);
102                 // 定义缓冲数组
103                 buffer = new byte[bufferSize];
104                 MAX_DATA_LENGTH = bufferSize / 2;
105                 audioRecord.startRecording(); // 开始录制
106                 isRecording = true; // 设置录制标记为true
107             }
108
109             @Override
110             protected void onPostExecute(String result) {
111                 audioRecord = null;
112                 if (result == null) {
113                     lastCacheFile = null;
```

```
114         } else {
115             lastCacheFile = new File(result);
116         }
117         if (audioRecordingCallback != null) {
118             audioRecordingCallback.onStopRecord(result);
119         }
120
121
122         5 @Override
123         protected String doInBackground(String... params) {
124             String tempFileName = null;
125
126             // 开始录制
127             while (isRecording) {
128                 // 录制的内容放置到了buffer中，result代表存储长度
129                 int result = audioRecord.read(buffer, 0, buffer.length);
130                 // 将数据回调出去
131                 if (audioRecordingCallback != null) {
132                     int offset = result % MAX_DATA_LENGTH > 0 ? 1 : 0;
133                     for (int i = 0; i < result / MAX_DATA_LENGTH + offset; i++) {
134                         int length = MAX_DATA_LENGTH;
135                         if ((i + 1) * MAX_DATA_LENGTH > result) {
136                             length = result - i * MAX_DATA_LENGTH;
137                         }
138                         audioRecordingCallback.onRecording(buffer, i
139                             * MAX_DATA_LENGTH, length);
140                     }
141                 }
142                 if (audioRecord != null) {
143                     audioRecord.stop();
144                 }
145                 return tempFileName;
146             }
147         }
148
149         /**
150          * 监听录制过程，用于实时获取录音数据
151          *
152          * @author 李长军
153          *
154          */
155         public static interface AudioRecordingCallback {
156             /**
157              * 录音数据获取回调
158              *
159              * @param data
160              *      数据数组对象
161              * @param startIndex
162              *      数据其开始
163              * @param length
164              *      数据的结尾
165              */
166             public void onRecording(byte[] data, int startIndex, int length);
167
168             /**
169              * 录音结束后的回调
170              *
```

```
171      * @param savedPath
172      *      录音文件存储的路径
173      */
174      public void onStopRecord(String savedPath);
175  }
176
177  /**
178   * 释放资源
179   */
180  public void release() {
181      if (audioRecord != null) {
182          audioRecord.release();
183          audioRecord = null;
184      }
185  }
186  }
187
188 }
```

【声音播放的源码】


```
1
2 /**
3  * 实时音频播放处理类<br/>
4  * 使用示例代码如下:<br/>
5  *
6  * <pre>
7  *  AudioPlayerHandler = new AudioPlayerHandler();
8  * audioPlayerHandler.prepare();// 播放前需要prepare。可以重复prepare
9  *  直接将需要播放的数据传入即可
10 *  audioPlayerHandler.onPlaying(data, 0, data.length);
11 * </pre>
12 *  ...
13 * @author 李长军
14 * 
15 */
16 @SuppressWarnings("deprecation")
17 public class AudioPlayerHandler implements Runnable {
18
19     private AudioTrack track = null;// 录音文件播放对象
20     private boolean isPlaying = false;// 标记是否正在录音中
21     private int frequency = 8000;// 采样率 8000
22     private int channelInConfig = AudioFormat.CHANNEL_OUT_MONO;// 定义采样为双声道（过时，但是使用其他的又不行
23     private int audioEncoding = AudioFormat.ENCODING_PCM_16BIT;// 定义音频编码（16位）
24     private int bufferSize = -1;// 播放缓冲大小
25     private LinkedBlockingDeque<Object> dataQueue = new LinkedBlockingDeque<>();
26     // 互斥信号量
27     private Semaphore semaphore = new Semaphore(1);
28     // 是否释放资源的标志位
29     private boolean release = false;
30
31     public AudioPlayerHandler() {
32         // 获取缓冲大小
33         bufferSize = AudioTrack.getMinBufferSize(frequency, channelInConfig,
34             audioEncoding);
35
36         // 实例AudioTrack
37         track = new AudioTrack(AudioManager.STREAM_MUSIC, frequency,
38             channelInConfig, audioEncoding, bufferSize,
39             AudioTrack.MODE_STREAM);
40         track.setStereoVolume(AudioTrack.getMaxVolume(),
41             AudioTrack.getMaxVolume());
42         try {
43             // 默认需要抢占一个信号量。防止播放进程执行
44             semaphore.acquire();
45         } catch (InterruptedException e) {
46             e.printStackTrace();
47         }
48         // 开启播放线程
49         new Thread(this).start();
50     }
51
52     /**
53     * 播放，当有新数据传入时，
54     *
55     * @param data
56     *     语音byte数组
```

```
57     * @param startIndex
58     *     开始的偏移量
59     * @param length
60     *     数据长度
61     */
62     public synchronized void onPlaying(byte[] data, int startIndex, int length) {
63         if (AudioTrack.ERROR_BAD_VALUE == bufferSize) { // 初始化错误
64             return;
65         }
66         try {
67             dataQueue.putLast(data);
68             semaphore.release();
69         } catch (InterruptedException e) {
70             e.printStackTrace();
71         }
72     }
73
74     /**
75     * 准备播放
76     */
77     public void prepare() {
78         if (track != null && !isPlaying) {
79             track.play();
80             isPlaying = true;
81         }
82     }
83
84
85     /**
86     * 停止播放
87     */
88     public void stop() {
89         if (track != null) {
90             track.stop();
91             isPlaying = false;
92         }
93     }
94
95     /**
96     * 释放资源
97     */
98     public void release() {
99         release = true;
100         semaphore.release();
101         if (track != null) {
102             track.release();
103             track = null;
104         }
105     }
106
107     @Override
108     public void run() {
109         while (true) {
110             if (release) {
111                 return;
112             }
113             if (dataQueue.size() > 0) {
```

```
114         byte[] data = (byte[]) dataQueue.pollFirst();
115         track.write(data, 0, data.length);
116     } else {
117         try {
118             semaphore.acquire();
119         } catch (InterruptedException e) {
120             e.printStackTrace();
121         }
122     }
123 }
124 }
125 }
```



版权声明：本文为博主原创文章，未经博主允许不得转载。



mqy1023 (/mqy1023) 2017-08-28 22:45

12楼

请问能提供吗？279371794@qq.com。谢谢！！

回复



openopen_119 (/openopen_119) 2017-08-20 18:39

11楼

谢谢楼主分享。我在用AudioTrack做实时音频播放的时候，播放出来的音频断断续续、卡顿、不连贯、噪音很大、延时比较高，大概1~2秒。楼主有没有好的办法，希望楼主能够给我提供点建议，非常感谢。这是我的qq：2466373520

回复



wwangcongshuai (/wwangcongshuai) 2017-08-09 18:24

10楼

想请教一个问题呢：就是我用AudioRecord进行录音操作的过程中（说了一句话然后等待但是不停止录音），等待过程中用AudioTrack播放了一段音频，之后我再说话，显示录到的音数据全为0，这是怎么回事啊？非常感谢！

回复

查看 14 条热评

相关文章推荐

【android开发】实现语音数据实时采集/播放 (http://blog.csdn.net/huang_rong12/articl...)

今天无意中看到一篇关于android实现语音数据实时采集/播放的文章，感觉写得非常棒，挺全面的，所以特地转载了，还有其实还可以根据这篇博客内容考虑下视频数据实时采集、播放的实现。博客原文地址<http://...>





huang_rong12 (http://blog.csdn.net/huang_rong12) 2016年12月09日 09:02

2011

研究**Android**即时聊天、实时语音通话、实时对讲机等必备知识**Audio** (<http://blog.cs...>)



Android深入浅出之**Audio** 第一部分 **AudioTrack**分析一目的本文的目的是通过从**Audio**系统来分析**Android**的代码，包括**Android**自定义的那套机制和一些常...

 d1020965026 (<http://blog.csdn.net/d1020965026>) 2014年05月06日 13:13  7440



Android实时语音聊天 (http://blog.csdn.net/lv_ws/article/details/41863149)



近些天一直在做利用**RTMP**实现实时语音聊天的**Android**聊天室，中间遇到不少问题，终于有时间记录一下。1.关于**RTMP** **RTMP**协议不必多说。<http://www.cnblogs.com/ha...>

 lv_ws (http://blog.csdn.net/lv_ws) 2014年12月11日 10:19  2177



Android 多媒体之实现语音聊天界面 (<http://blog.csdn.net/lovoo/article/details/51661077>)

一、概述：1、要实现的功能：1) 语音的录制 2) 按钮状态的切换 3) 对话框状态的切换 4) 聊天界面的切换 2、所用的技术：1) 核心技术：根据按钮的**onTouchEvent**实现...

 lovoo (<http://blog.csdn.net/lovoo>) 2016年06月13日 22:19  2589



Delphi7高级应用开发随书源码 (<http://download.csdn.net/detail/chexh...>)

<http://download.csdn.net/detail/chexh...> 2003年04月30日 00:00 676KB [下载](#) ([...](#))





Delphi7高级应用开发随书源码 (<http://download.csdn.net/detail/chexh...>)

<http://download.csdn.net/detail/chexh...> 2003年04月30日 00:00 676KB [下载](#) ([...](#))



Android中播放声音的两种方法 (http://blog.csdn.net/pku_android/article/details/7625868)

作者：高天辰 在**Android**中，音频、视频等多媒体元素的加入，使得应用程序的用户体验更好。可以说，现在的手机，已经远远不只作为通信工具，更成为娱乐、办公的必备产品。...

 pku_android (http://blog.csdn.net/pku_android) 2012年06月02日 14:10  75194



Android 语音播放**Media Player** (<http://blog.csdn.net/yy254117440/article/details/52945449>)

原文地址：<https://developer.android.com/guide/topics/media/mediaplayer.html#viacontentresolver>语音播放因为实习工...

 yy254117440 (<http://blog.csdn.net/yy254117440>) 2016年10月27日 15:24  1313



Android 音频 录音与播放 (<http://blog.csdn.net/wenxustqiang/article/details/71170163>)

声明：此文为本人整理网络资料加自己的一些注解所得。参考：<http://www.2cto.com/kf/201610/553744.html> <http://blog.csdn...>

 wenxustqiang (<http://blog.csdn.net/wenxustqiang>) 2017年05月04日 14:23  244

android播放网络音频 (http://blog.csdn.net/qq_19067845/article/details/51355021)



android播放网络音频，很简单的技术，但是可以学习下很简单的一个获取网络音频播放器，有进度条，播放，暂停，停止，重新播放，支持缓存，以下是源码，希望可以帮到大家布局文件很简单，就几个按钮，Text...

 qq_19067845 (http://blog.csdn.net/qq_19067845) 2016年05月09日 18:13  1954



Android实现录音功能及播放语音功能 (http://blog.csdn.net/ming_147/article/details/523...)



Android中实现录音功能其实很简单，直接调用的系统的就ok了，这里就不写实现的原理了，直接部署代码：所谓的实现就是用的MediaRecorder。录音功能代码： //开始录制 p...

 ming_147 (http://blog.csdn.net/ming_147) 2016年08月27日 16:01  1146





Android 音频采集（原始音频） (http://blog.csdn.net/Java_android_c/article/details/526...)

android平台上的音频采集一般就三种：1.利用android内置的应用程序 2.使用MediaRecorder进行音频捕获 3.使用audioRecord进行音频捕获。此3种的灵活性逐渐增...

 Java_android_c (http://blog.csdn.net/Java_android_c) 2016年09月22日 10:37  3310



Android MIC 口的音频采集 (<http://blog.csdn.net/shen809873995/article/details/44560757>)

本文主要针对谈论如何采集音频流，用MIC口的音频流采集做实例，以及一些采集过程中需要注意到的事项。...

 shen809873995 (<http://blog.csdn.net/shen809873995>) 2015年03月23日 10:58  1212



android视频和音频采集及预览 (<http://blog.csdn.net/zhuweigangzgw/article/details/5270...>)

android视频和音频采集及预览本文说明android采集方面的方法，预览用previcw，视频采集用Camera类，音频采集用AudioRecord，如果需要ios采集在我的博客中看上一篇，ff...

 zhuweigangzgw (<http://blog.csdn.net/zhuweigangzgw>) 2016年09月29日 14:49  2048



Android 音视频采集与软编码总结 (<http://blog.csdn.net/mabeijianxi/article/details/75807...>)

前言本文总结了笔者在 Android 音视频采集与软编码中的一些经验与技巧，包括移植 FFmpeg、YUV 视频帧处理、最新的 JNI 编写技巧、ndk 开发技巧等，为了不扯太远本文不会对视听...

 mabeijianxi (<http://blog.csdn.net/mabeijianxi>) 2017年07月22日 21:06  2273



Android音频采集、压缩、发送 (http://blog.csdn.net/gjy_it/article/details/56668019)

基本的实现流程： 1、从手机麦中采集音频数据；2、将PCM音频数据编码压缩；3、将压缩好的音频通过无线网络发送出去；4、其他手机接收音频数据并解码；5、将音频数据写入到音...

 gjy_it (http://blog.csdn.net/gjy_it) 2017年02月23日 11:52  1899



Android音频开发（2）：如何采集一帧音频 (<http://blog.csdn.net/u010880786/article/de...>)

本文重点关注如何在Android平台上采集一帧音频数据。阅读本文之前，建议先读一下我的上一篇文章《Android音频开发（1）：基础知识》，因为音频开发过程中，经常要涉及到这些基础知识，掌握了这些重要...

 u010880786 (<http://blog.csdn.net/u010880786>) 2016年05月28日 10:08  2807

android语音识别技术 (<http://blog.csdn.net/wangkuifeng0118/article/details/7251813>)

今天从网上找了个例子实现了语音识别，个人感觉挺好玩的，就把代码贴出来与大家分享下： Android中主要通过 RecognizerIntent来实现语音识别，其实代码比较简单，但是如...

 wangkuifeng0118 (<http://blog.csdn.net/wangkuifeng0118>) 2012年02月11日 23:35  53633



Android Studio下Android应用开发集成百度语音合成使用方法样例 (<http://blog.csdn.net/kjunchen/article/details/51093134>)



转载请注明来源: <http://blog.csdn.net/kjunchen/article/details/51093134> Android Studio下Android应用开发集成百度语音合成使用...

 KjunChen (<http://blog.csdn.net/KjunChen>) 2016年04月08日 07:51  9080



android开发语音播报 (<http://blog.csdn.net/sw950729/article/details/52069582>)

转载请注明出处: <http://blog.csdn.net/sw950729/article/details/52050777> 本文出自: 马云飞的博客上次也说了最近一直在整理东西以及封装东西。然...

 sw950729 (<http://blog.csdn.net/sw950729>) 2016年07月30日 00:09  4955