

Google、在奋斗

我只有很努力、你看我才觉毫不费力。

目录视图

摘要视图

订阅

个人资料

Google_acmer

访问：2574次

积分：423分

排名：千里之外

原创：40篇 转载：2篇

译文：0篇 评论：5条

公告栏

风过留痕
雁过留声
如果您真的喜欢我的文章
可以留下评论
我们一起讨论
希望对你的学习有所帮助
转载请注明出处、谢谢您的浏览

文章搜索

博客专栏

GL音乐播放器开发

文章：12篇

阅读：1726

文章分类

Android开发小知识积累系列 (16)

android开发-GL音乐播放器系列 (11)

ACM--字符串 (1)

ACM--省赛集 (1)

ACM--图论 (1)

ACM--排序 (1)

心情随笔 (3)

人生感悟 (3)

博客始记 (1)

【大声说出你的爱】CSDN社区情人节特别活动

专访李云：从通讯行业的架构师到互联网“新兵”

电子版《程序员》杂志免费领取

GL音乐播放器<三>--界面设计之专辑照片的实现

分类： android开发-GL音乐播放器系列

2014-02-18 12:49

213人阅读

评论(0)

收藏

举报

这篇博客我会向大家介绍如何实现歌曲专辑照片的实现。但是注意，只是歌曲自带的照片（科普一下，一般正版的音乐里都会带有这首歌所在专辑的专辑照片），我们接下来要做的就是找到这个照片，并将它显示出来。

先上效果图：

这就是我们要实现的效果。

首先是要获得专辑封面的Uri，得到并进行处理

```
[java] view plain copy print ?
01. package com.genius.musicplay;
02.
03. import java.io.FileDescriptor;
04. import java.io.FileNotFoundException;
05. import java.io.IOException;
06. import java.io.InputStream;
07.
08. import com.genius.demo.R;
09.
10.
11.
12. import android.content.ContentResolver;
13. import android.content.ContentUri;
14. import android.content.Context;
15. import android.graphics.Bitmap;
16. import android.graphics.BitmapFactory;
17. import android.graphics.BitmapFactory.Options;
18. import android.net.Uri;
19. import android.os.ParcelFileDescriptor;
20.
21. public class MediaAlbum {
22.     //获取专辑封面的Uri
23.     private static final Uri albumArtUri = Uri.parse("content://media/external/audio/
24.     /**
25.     * 获取默认专辑图片
```

大连海事大学第一届编程马拉松挑战赛多校联赛DIV1 第一场 (3)

文章存档

2014年02月 (42)

阅读排行

GL音乐播放器--代码实现 (239)

GL音乐播放器<三>--界面 (212)

GL音乐播放器--代码实现 (206)

Android 广播大全 Intent (193)

GL音乐播放器--代码实现 (158)

GL音乐播放器--代码实现 (155)

GL音乐播放器--代码实现 (146)

GL音乐播放器1.0.0版--总结帖 (138)

GL音乐播放器<二>----界面 (129)

GL音乐播放器--代码实现 (119)

评论排行

GL音乐播放器--代码实现 (3)

GL音乐播放器--代码实现 (1)

GL音乐播放器1.0.0版--总结帖 (0)

Android开发环境搭建教程 (0)

GL音乐播放器--代码实现 (0)

心情随笔---<二> (0)

关于向安卓模拟机SD卡写入数据 (0)

GL音乐播放器---代码实现 (0)

GL音乐播放器<二>----界面 (0)

visibility属性 (0)

推荐文章

* JBoss 系列九十三： 高性能非阻塞 Web 服务器 Undertow

* iOS安全攻防（十九）： 基于脚本实现动态库注入

* Java7之基础 - 强引用、弱引用、软引用、虚引用

* 深入理解SELinux/SEAndroid（最后部分）

* Android屏幕适配解析 - 详解像素、设备独立像素、归一化密度、精确密度及各种资源对应的尺寸密度分辨率适配问题

* 快速检测空间三角形相交算法的代码实现 (Devillers & Guigue算法)

最新评论

GL音乐播放器1.0.0版--总结帖 卡夫卡卡: 积分已献~小表支持

GL音乐播放器--代码实现<六>----界面 卡夫卡卡: 不错的系列，一直在关注。哈哈

GL音乐播放器--代码实现<四>----界面 Google_acmer: @u013295109: 对，我刚开始就是看的他的代码，感觉那个抽屉挺好玩，学习了好多但是感觉他有好多...

GL音乐播放器--代码实现<四>----界面 Google_acmer: @u013295109: 基本做完了，以后也就是加点功能什么的。

```
26.      * @param context
27.      * @return
28.      */
29.      public static Bitmap getDefaultArtwork(Context context,boolean small) {
30.          BitmapFactory.Options opts = new BitmapFactory.Options();
31.          opts.inPreferredConfig = Bitmap.Config.RGB_565;
32.          if(small){ //返回小图片
33.              return BitmapFactory.decodeStream(context.getResources().openRawResource(R.drawable.albumart), null, opts);
34.          }
35.          return BitmapFactory.decodeStream(context.getResources().openRawResource(R.drawable.albumart), null, opts);
36.      }
37.
38.
39.      /**
40.       * 从文件当中获取专辑封面位图
41.       * @param context
42.       * @param songid
43.       * @param albumid
44.       * @return
45.       */
46.      private static Bitmap getArtworkFromFile(Context context, long songid, long albumid) {
47.          Bitmap bm = null;
48.          if(albumid < 0 && songid < 0) {
49.              throw new IllegalArgumentException("Must specify an album or a song id");
50.          }
51.          try {
52.              BitmapFactory.Options options = new BitmapFactory.Options();
53.              FileDescriptor fd = null;
54.              if(albumid < 0){
55.                  Uri uri = Uri.parse("content://media/external/audio/media/" + songid + "/" + "albumart");
56.                  ParcelFileDescriptor pfd = context.getContentResolver().openFileDescriptor(uri, "r");
57.                  if(pfd != null) {
58.                      fd = pfd.getFileDescriptor();
59.                  }
60.              } else {
61.                  Uri uri = ContentUris.withAppendedId(albumArtUri, albumid);
62.                  ParcelFileDescriptor pfd = context.getContentResolver().openFileDescriptor(uri, "r");
63.                  if(pfd != null) {
64.                      fd = pfd.getFileDescriptor();
65.                  }
66.              }
67.              options.inSampleSize = 1;
68.              // 只进行大小判断
69.              options.inJustDecodeBounds = true;
70.              // 调用此方法得到options得到图片大小
71.              BitmapFactory.decodeFileDescriptor(fd, null, options);
72.              // 我们的目标是在800pixel的画面上显示
73.              // 所以需要调用computeSampleSize得到图片缩放的比例
74.              options.inSampleSize = 100;
75.              // 我们得到了缩放的比例，现在开始正式读入Bitmap数据
76.              options.inJustDecodeBounds = false;
77.              options.inDither = false;
78.              options.inPreferredConfig = Bitmap.Config.ARGB_8888;
79.
80.              //根据options参数，减少所需要的内存
81.              bm = BitmapFactory.decodeFileDescriptor(fd, null, options);
82.          } catch (FileNotFoundException e) {
83.              e.printStackTrace();
84.          }
85.          return bm;
86.      }
87.
88.
89.      /**
90.       * 获取专辑封面位图对象
91.       * @param context
92.       * @param song_id
93.       * @param album_id
94.       * @param allowdefault
95.       * @return
96.       */
97.      public static Bitmap getArtwork(Context context, long song_id, int album_id, boolean allowdefault) {
98.          if(album_id < 0) {
99.              if(song_id < 0) {
100.                  Bitmap bm = getArtworkFromFile(context, song_id, -1);
101.                  if(bm != null) {
102.                      return bm;
```

GL音乐播放器--代码实现<四>--过卡夫卡卡: 楼主这个播放器做完了吗

```

103.         }
104.     }
105.     if(allowdefalut) {
106.         return getDefaultArtwork(context, small);
107.     }
108.     return null;
109. }
110. ContentResolver res = context.getContentResolver();
111. Uri uri = ContentUris.withAppendedId(albumArtUri, album_id);
112. if(uri != null) {
113.     InputStream in = null;
114.     try {
115.         in = res.openInputStream(uri);
116.         BitmapFactory.Options options = new BitmapFactory.Options();
117.         //先制定原始大小
118.         options.inSampleSize = 1;
119.         //只进行大小判断
120.         options.inJustDecodeBounds = true;
121.         //调用此方法得到options得到图片的大小
122.         BitmapFactory.decodeStream(in, null, options);
123.         /** 我们的目标是在你N pixel的画面上显示。 所以需要调用computeSampleSize得
到图片缩放的比例 **/
124.         /** 这里的target为800是根据默认专辑图片大小决定的, 800只是测试数字但是试验后
发现完美的结合 **/
125.         if(small){
126.             options.inSampleSize = computeSampleSize(options, 40);
127.         } else{
128.             options.inSampleSize = computeSampleSize(options, 600);
129.         }
130.         // 我们得到了缩放比例, 现在开始正式读入Bitmap数据
131.         options.inJustDecodeBounds = false;
132.         options.inDither = false;
133.         options.inPreferredConfig = Bitmap.Config.ARGB_8888;
134.         in = res.openInputStream(uri);
135.         return BitmapFactory.decodeStream(in, null, options);
136.     } catch (FileNotFoundException e) {
137.         Bitmap bm = getArtworkFromFile(context, song_id, album_id);
138.         if(bm != null) {
139.             if(bm.getConfig() == null) {
140.                 bm = bm.copy(Bitmap.Config.RGB_565, false);
141.                 if(bm == null && allowdefalut) {
142.                     return getDefaultArtwork(context, small);
143.                 }
144.             }
145.         } else if(allowdefalut) {
146.             bm = getDefaultArtwork(context, small);
147.         }
148.         return bm;
149.     } finally {
150.         try {
151.             if(in != null) {
152.                 in.close();
153.             }
154.         } catch (IOException e) {
155.             e.printStackTrace();
156.         }
157.     }
158. }
159. return null;
160. }
161.
162. /**
163.  * 对图片进行合适的缩放
164.  * @param options
165.  * @param target
166.  * @return
167.  */
168. public static int computeSampleSize(Options options, int target) {
169.     int w = options.outWidth;
170.     int h = options.outHeight;
171.     int candidateW = w / target;
172.     int candidateH = h / target;
173.     int candidate = Math.max(candidateW, candidateH);
174.     if(candidate == 0) {
175.         return 1;
176.     }
177.     if(candidate > 1) {
178.         if((w > target) && (w / candidate) < target) {
179.             candidate -= 1;

```

```
180.         }
181.     }
182.     if(candidate > 1) {
183.         if((h > target) && (h / candidate) < target) {
184.             candidate -= 1;
185.         }
186.     }
187.     return candidate;
188. }
189. }
```

接下来是图片处理的工具类，这个是别人已经完成的，所以我就偷懒了，嘻嘻

```
[java] view plain copy print ? C ?
01. package com.genius.widget;
02.
03. import java.io.ByteArrayOutputStream;
04. import java.io.File;
05. import java.io.FileNotFoundException;
06. import java.io.FileOutputStream;
07. import java.io.IOException;
08.
09. import android.content.Context;
10. import android.graphics.Bitmap;
11. import android.graphics.Bitmap.CompressFormat;
12. import android.graphics.Bitmap.Config;
13. import android.graphics.BitmapFactory;
14. import android.graphics.Canvas;
15. import android.graphics.ColorMatrix;
16. import android.graphics.ColorMatrixColorFilter;
17. import android.graphics.LinearGradient;
18. import android.graphics.Matrix;
19. import android.graphics.Paint;
20. import android.graphics.PorterDuff;
21. import android.graphics.PorterDuff.Mode;
22. import android.graphics.PorterDuffXfermode;
23. import android.graphics.Rect;
24. import android.graphics.RectF;
25. import android.graphics.Shader;
26. import android.graphics.drawable.BitmapDrawable;
27. import android.graphics.drawable.Drawable;
28.
29. /**
30.  * 图片工具类
31.  *
32.  */
33. public class ImageUtil {
34.     /**图片的八个位置**/
35.     public static final int TOP = 0;           //上
36.     public static final int BOTTOM = 1;        //下
37.     public static final int LEFT = 2;         //左
38.     public static final int RIGHT = 3;        //右
39.     public static final int LEFT_TOP = 4;     //左上
40.     public static final int LEFT_BOTTOM = 5;  //左下
41.     public static final int RIGHT_TOP = 6;    //右上
42.     public static final int RIGHT_BOTTOM = 7; //右下
43.
44.     /**
45.      * 图像的放大缩小方法
46.      * @param src      源位图对象
47.      * @param scaleX   宽度比例系数
48.      * @param scaleY   高度比例系数
49.      * @return 返回位图对象
50.      */
51.     public static Bitmap zoomBitmap(Bitmap src, float scaleX, float scaleY) {
52.         Matrix matrix = new Matrix();
53.         matrix.setScale(scaleX, scaleY);
54.         Bitmap t_bitmap = Bitmap.createBitmap(src, 0, 0, src.getWidth(), src.getHeight(),
55.             matrix, true);
56.         return t_bitmap;
57.     }
58.
59.     /**
60.      * 图像放大缩小--根据宽度和高度
61.      * @param src
```

```
61.         * @param width
62.         * @param height
63.         * @return
64.         */
65.     public static Bitmap zoomBimtap(Bitmap src, int width, int height) {
66.         return Bitmap.createScaledBitmap(src, width, height, true);
67.     }
68.
69.     /**
70.      * 将Drawable转为Bitmap对象
71.      * @param drawable
72.      * @return
73.      */
74.     public static Bitmap drawableToBitmap(Drawable drawable) {
75.         return ((BitmapDrawable)drawable).getBitmap();
76.     }
77.
78.
79.     /**
80.      * 将Bitmap转换为Drawable对象
81.      * @param bitmap
82.      * @return
83.      */
84.     public static Drawable bitmapToDrawable(Bitmap bitmap) {
85.         Drawable drawable = new BitmapDrawable(bitmap);
86.         return drawable;
87.     }
88.
89.     /**
90.      * Bitmap转byte[]
91.      * @param bitmap
92.      * @return
93.      */
94.     public static byte[] bitmapToByte(Bitmap bitmap) {
95.         ByteArrayOutputStream out = new ByteArrayOutputStream();
96.         bitmap.compress(Bitmap.CompressFormat.PNG, 100, out);
97.         return out.toByteArray();
98.     }
99.
100.    /**
101.     * byte[]转Bitmap
102.     * @param data
103.     * @return
104.     */
105.    public static Bitmap byteToBitmap(byte[] data) {
106.        if(data.length != 0) {
107.            return BitmapFactory.decodeByteArray(data, 0, data.length);
108.        }
109.        return null;
110.    }
111.
112.    /**
113.     * 绘制带圆角的图像
114.     * @param src
115.     * @param radius
116.     * @return
117.     */
118.    public static Bitmap createRoundedCornerBitmap(Bitmap src, int radius) {
119.        final int w = src.getWidth();
120.        final int h = src.getHeight();
121.        // 高清图32位图
122.        Bitmap bitmap = Bitmap.createBitmap(w, h, Config.ARGB_8888);
123.        Paint paint = new Paint();
124.        Canvas canvas = new Canvas(bitmap);
125.        canvas.drawARGB(0, 0, 0, 0);
126.        paint.setColor(0xff424242);
127.        // 防止边缘的锯齿
128.        paint.setFilterBitmap(true);
129.        Rect rect = new Rect(0, 0, w, h);
130.        RectF rectf = new RectF(rect);
131.        // 绘制带圆角的矩形
132.        canvas.drawRoundRect(rectf, radius, radius, paint);
133.
134.        // 取两层绘制交集, 显示上层
135.        paint.setXfermode(new PorterDuffXfermode(PorterDuff.Mode.SRC_IN));
136.        // 绘制图像
137.        canvas.drawBitmap(src, rect, rect, paint);
138.        return bitmap;
139.    }
```

```
140.
141.     /**
142.     * 创建选中带提示图片
143.     * @param context
144.     * @param srcId
145.     * @param tipId
146.     * @return
147.     */
148.     public static Drawable createSelectedTip(Context context, int srcId, int tipId) {
149.         Bitmap src = BitmapFactory.decodeResource(context.getResources(), srcId);
150.         Bitmap tip = BitmapFactory.decodeResource(context.getResources(), tipId);
151.         final int w = src.getWidth();
152.         final int h = src.getHeight();
153.         Bitmap bitmap = Bitmap.createBitmap(w, h, Config.ARGB_8888);
154.         Paint paint = new Paint();
155.         Canvas canvas = new Canvas(bitmap);
156.         //绘制原图
157.         canvas.drawBitmap(src, 0, 0, paint);
158.         //绘制提示图片
159.         canvas.drawBitmap(tip, (w - tip.getWidth()), 0, paint);
160.         return bitmapToDrawable(bitmap);
161.     }
162.
163.     /**
164.     * 带倒影的图像
165.     * @param src
166.     * @return
167.     */
168.     public static Bitmap createReflectionBitmap(Bitmap src) {
169.         // 两个图像间的空隙
170.         final int spacing = 4;
171.         final int w = src.getWidth();
172.         final int h = src.getHeight();
173.         // 绘制高质量32位图
174.         Bitmap bitmap = Bitmap.createBitmap(w, h + h / 2 + spacing, Config.ARGB_8888);
175.         // 创建沿X轴的倒影图像
176.         Matrix m = new Matrix();
177.         m.setScale(1, -1);
178.         Bitmap t_bitmap = Bitmap.createBitmap(src, 0, h / 2, w, h / 2, m, true);
179.
180.         Canvas canvas = new Canvas(bitmap);
181.         Paint paint = new Paint();
182.         // 绘制原图像
183.         canvas.drawBitmap(src, 0, 0, paint);
184.         // 绘制倒影图像
185.         canvas.drawBitmap(t_bitmap, 0, h + spacing, paint);
186.         // 线性渲染-沿Y轴高到低渲染
187.         Shader shader = new LinearGradient(0, h + spacing, 0, h + spacing + h / 2, 0x70ff,
188.             paint.setShader(shader);
189.         // 取两层绘制交集, 显示下层。
190.         paint.setXfermode(new PorterDuffXfermode(Mode.DST_IN));
191.         // 绘制渲染倒影的矩形
192.         canvas.drawRect(0, h + spacing, w, h + h / 2 + spacing, paint);
193.         return bitmap;
194.     }
195.
196.
197.     /**
198.     * 独立的倒影图像
199.     * @param src
200.     * @return
201.     */
202.     public static Bitmap createReflectionBitmapForSingle(Bitmap src) {
203.         final int w = src.getWidth();
204.         final int h = src.getHeight();
205.         // 绘制高质量32位图
206.         Bitmap bitmap = Bitmap.createBitmap(w, h / 2, Config.ARGB_8888);
207.         // 创建沿X轴的倒影图像
208.         Matrix m = new Matrix();
209.         m.setScale(1, -1);
210.         Bitmap t_bitmap = Bitmap.createBitmap(src, 0, h / 2, w, h / 2, m, true);
211.
212.         Canvas canvas = new Canvas(bitmap);
213.         Paint paint = new Paint();
214.         // 绘制倒影图像
215.         canvas.drawBitmap(t_bitmap, 0, 0, paint);
216.         // 线性渲染-沿Y轴高到低渲染
217.         Shader shader = new LinearGradient(0, 0, 0, h / 2, 0x70ffffff,
218.             0x00ffffff, Shader.TileMode.MIRROR);
```

```

219.         paint.setShader(shader);
220.         // 取两层绘制交集。显示下层。
221.         paint.setXfermode(new PorterDuffXfermode(Mode.DST_IN));
222.         // 绘制渲染倒影的矩形
223.         canvas.drawRect(0, 0, w, h / 2, paint);
224.         return bitmap;
225.     }
226.
227.
228.     public static Bitmap createGreyBitmap(Bitmap src) {
229.         final int w = src.getWidth();
230.         final int h = src.getHeight();
231.         Bitmap bitmap = Bitmap.createBitmap(w, h, Config.ARGB_8888);
232.         Canvas canvas = new Canvas(bitmap);
233.         Paint paint = new Paint();
234.         // 颜色变换的矩阵
235.         ColorMatrix matrix = new ColorMatrix();
236.         // saturation 饱和度值，最小可设为0，此时对应的是灰度图；为1表示饱和度不变，设置大于1，就
显示过饱和
237.         matrix.setSaturation(0);
238.         ColorMatrixColorFilter filter = new ColorMatrixColorFilter(matrix);
239.         paint.setColorFilter(filter);
240.         canvas.drawBitmap(src, 0, 0, paint);
241.         return bitmap;
242.     }
243.
244.     /**
245.      * 保存图片
246.      * @param src
247.      * @param filepath
248.      * @param format:[Bitmap.CompressFormat.PNG,Bitmap.CompressFormat.JPEG]
249.      * @return
250.      */
251.     public static boolean saveImage(Bitmap src, String filepath, CompressFormat format) {
252.         boolean rs = false;
253.         File file = new File(filepath);
254.         try {
255.             FileOutputStream out = new FileOutputStream(file);
256.             if(src.compress(format, 100, out)) {
257.                 out.flush(); //写入流
258.             }
259.             out.close();
260.         } catch (FileNotFoundException e) {
261.             e.printStackTrace();
262.         } catch (IOException e) {
263.             e.printStackTrace();
264.         }
265.         return rs;
266.     }
267.
268.     /**
269.      * 添加水印效果
270.      * @param src 源位图
271.      * @param watermark 水印
272.      * @param direction 方向
273.      * @param spacing 间距
274.      * @return
275.      */
276.     public static Bitmap createWatermark(Bitmap src, Bitmap watermark, int direction, int
277.         final int w = src.getWidth();
278.         final int h = src.getHeight();
279.         Bitmap bitmap = Bitmap.createBitmap(w, h, Config.ARGB_8888);
280.         Canvas canvas = new Canvas(bitmap);
281.         canvas.drawBitmap(src, 0, 0, null);
282.         if(direction == LEFT_TOP) {
283.             canvas.drawBitmap(watermark, spacing, spacing, null);
284.         } else if(direction == LEFT_BOTTOM){
285.             canvas.drawBitmap(watermark, spacing, h - watermark.getHeight() - spacing, null);
286.         } else if(direction == RIGHT_TOP) {
287.             canvas.drawBitmap(watermark, w - watermark.getWidth() - spacing, spacing, null);
288.         } else if(direction == RIGHT_BOTTOM) {
289.             canvas.drawBitmap(watermark, w - watermark.getWidth() - spacing, h - watermark
290.         }
291.         return bitmap;
292.     }
293.
294.
295.     /**
296.      * 合成图像

```

```

297.     * @param direction
298.     * @param bitmaps
299.     * @return
300.     */
301.     public static Bitmap composeBitmap(int direction, Bitmap... bitmaps) {
302.         if(bitmaps.length < 2) {
303.             return null;
304.         }
305.         Bitmap firstBitmap = bitmaps[0];
306.         for (int i = 0; i < bitmaps.length; i++) {
307.             firstBitmap = composeBitmap(firstBitmap, bitmaps[i], direction);
308.         }
309.         return firstBitmap;
310.     }
311.
312.     /**
313.     * 合成两张图像
314.     * @param firstBitmap
315.     * @param secondBitmap
316.     * @param direction
317.     * @return
318.     */
319.     private static Bitmap composeBitmap(Bitmap firstBitmap, Bitmap secondBitmap,
320.         int direction) {
321.         if(firstBitmap == null) {
322.             return null;
323.         }
324.         if(secondBitmap == null) {
325.             return firstBitmap;
326.         }
327.         final int fw = firstBitmap.getWidth();
328.         final int fh = firstBitmap.getHeight();
329.         final int sw = secondBitmap.getWidth();
330.         final int sh = secondBitmap.getHeight();
331.         Bitmap bitmap = null;
332.         Canvas canvas = null;
333.         if(direction == TOP) {
334.             bitmap = Bitmap.createBitmap(sw > fw ? sw : fw, fh + sh, Config.ARGB_8888);
335.             canvas = new Canvas(bitmap);
336.             canvas.drawBitmap(secondBitmap, 0, 0, null);
337.             canvas.drawBitmap(firstBitmap, 0, sh, null);
338.         } else if(direction == BOTTOM) {
339.             bitmap = Bitmap.createBitmap(fw > sw ? fw : sw, fh + sh, Config.ARGB_8888);
340.             canvas = new Canvas(bitmap);
341.             canvas.drawBitmap(firstBitmap, 0, 0, null);
342.             canvas.drawBitmap(secondBitmap, 0, fh, null);
343.         } else if(direction == LEFT) {
344.             bitmap = Bitmap.createBitmap(fw + sw, sh > fh ? sh : fh, Config.ARGB_8888);
345.             canvas = new Canvas(bitmap);
346.             canvas.drawBitmap(secondBitmap, 0, 0, null);
347.             canvas.drawBitmap(firstBitmap, sw, 0, null);
348.         } else if(direction == RIGHT) {
349.             bitmap = Bitmap.createBitmap(fw + sw, fh > sh ? fh : sh,
350.                 Config.ARGB_8888);
351.             canvas = new Canvas(bitmap);
352.             canvas.drawBitmap(firstBitmap, 0, 0, null);
353.             canvas.drawBitmap(secondBitmap, fw, 0, null);
354.         }
355.         return bitmap;
356.     }
357.
358.
359. }

```

接下来便是在MusicPlayActivity里创建一个用来显示图片的函数

```

[java] view plain copy print ?
01. private ImageView musicAlbum; //音乐专辑封面
02. private ImageView musicAlbumReflection; //倒影反射

[java] view plain copy print ?
01. //显示专辑照片
02. private void showArtwork(MusicData mp3Info) {

```



```
03.         Bitmap bm = MediaAlbum.getArtwork(this, mp3Info.MusicId, mp3Info.MusicAlbumId, tr
04.         //切换播放时候专辑图片出现透明效果
05.         Animation albumanim = AnimationUtils.loadAnimation(MusicPlayActivity.this
06.         //开始播放动画效果
07.         musicAlbum.startAnimation(albumanim);
08.         if(bm != null) {
09.             musicAlbum.setImageBitmap(bm); //显示专辑封面图片
10.             musicAblumReflection.setImageBitmap(ImageUtil.createReflectionBitmapF
显示倒影
11.         } else {
12.             bm = MediaAlbum.getDefaultArtwork(this, false);
13.             musicAlbum.setImageBitmap(bm); //显示专辑封面图片
14.             musicAblumReflection.setImageBitmap(ImageUtil.createReflectionBitmapF
显示倒影
15.         }
16.     }
```

然后每次更新歌曲后调用就可以了。

更多 0

上一篇：[GL音乐播放器--代码实现<三>](#)

下一篇：[GL音乐播放器--代码实现<四>--进度控制和可隐藏的音量控制](#)

顶 0 踩 0

相关主题推荐 界面设计 音乐 图片处理 32位 imageview

相关博文推荐

- 【android学习】imageview... OpenCV-2.4.8+VS2010+...
- WinSock网络编程学习笔记(四)网络... 【分享】50000句中文微博句法树库（样...
- 巴西小镇拍到狼人影像 APP设计流程
- 如何设计好一款iOS游戏UI的细节 Protect Mode



400-654-7778



iOS开发培训
先就业后付款，就业年薪 十万起步
零基础入学，不就业退全款


零学费入学

查看评论

暂无评论

发表评论

用户名: u013295109
评论内容:



提交

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

专区推荐内容

企业安全技术（1）：检查平台AE...

使用非html5实现js...

跟燕青一起学Android应用开...

苹果的顶级应用程序：设计，语言，...

2013年热门Android应用

如何在Windows下编译Open...

<< >>

更多招聘职位

【北京极游互动时代科技有限公司】c#

【北京极游互动时代科技有限公司】co

【啪啪】前端工程师

【广州聚多网络科技有限公司】iOS开

【深圳开饭喇信息科技有限公司（Openl

发）

核心技术类目

全部主题

Java

VPN

Android

iOS

ERP

IE10

Eclipse

CRM

JavaScript

Ubuntu

NFC

WAP

jQuery

数据库

BI

HTML5

Spring

Apache

Hadoop

.NET

API

HTML

SDK

IIS

Fedora

XML

LBS

Unity

Splashtop

UML

components

Windows Mobile

Rails

QEMU

KDE

Cassandra

CloudStack

FTC

coremail

OPhone

CouchBase

云计算

iOS6

Rackspace

Web App

SpringSide

Maemo

Compuware

大数据

aptech

Perl

Tornado

Ruby

Hibernate

ThinkPHP

Spark

HBase

Pure

Solr

Angular

Cloud Foundry

Redis

Scala

Django

Bootstrap

公司简介 | 招贤纳士 | 广告服务 | 银行汇款帐号 | 联系方式 | 版权声明 | 法律顾问 | 问题报告

QQ客服 微博客服 论坛反馈 联系邮箱: webmaster@csdn.net 服务热线: 400-600-2320

京 ICP 证 070598 号

北京创新乐知信息技术有限公司 版权所有

江苏乐知网络技术有限公司 提供商务支持

Copyright © 1999-2014, CSDN.NET, All Rights Reserved

