



南方科技大学  
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

# 《电子科学创新实验 I》 课程报告

题    目： 基于 Android Studio 的  
            音乐播放器 APP 的设计与开发

姓    名： 张皓东

学    号： 12113010

系    别： 电子与电气工程系

专    业： 通信工程

指导教师： 徐琳琳

## 基于 Android Studio 的音乐播放器 APP 开发

[摘要]: Android Studio 是谷歌推出的一个 Android 集成开发工具, 基于 IntelliJ IDEA 用于 Android APP 的开发。Android Studio 提供了集成的 Android 开发工具用于开发和调试。本项目用 Android Studio 设计了一款音乐播放器 APP, 实现了以下功能: 1.使用 RecyclerView 来得到歌曲列表。2.具有音乐的播放、暂停、下一首、上一首、单曲循环和列表循环的选择等功能。3.具有进度条的计时功能。4.可以在歌曲列表选择歌曲, 跳转到相应歌曲播放界面, 并可以点击按钮回到歌曲列表。5.对不同的歌曲会自动切换不同的背景图案和歌曲图案, 进行了界面美化。在上述功能实现的同时该项目针对大量的细节问题进行了修复和优化, 从而使得该项目更加具有实用性和人机友好性, 便于用户使用。

# 1. 项目介绍和详解与原项目的比较

## 1.1 界面优化

本项目对原有项目的界面进行了较大的改进和优化。首先先展示前后的界面效果展示对比：

原始项目界面：

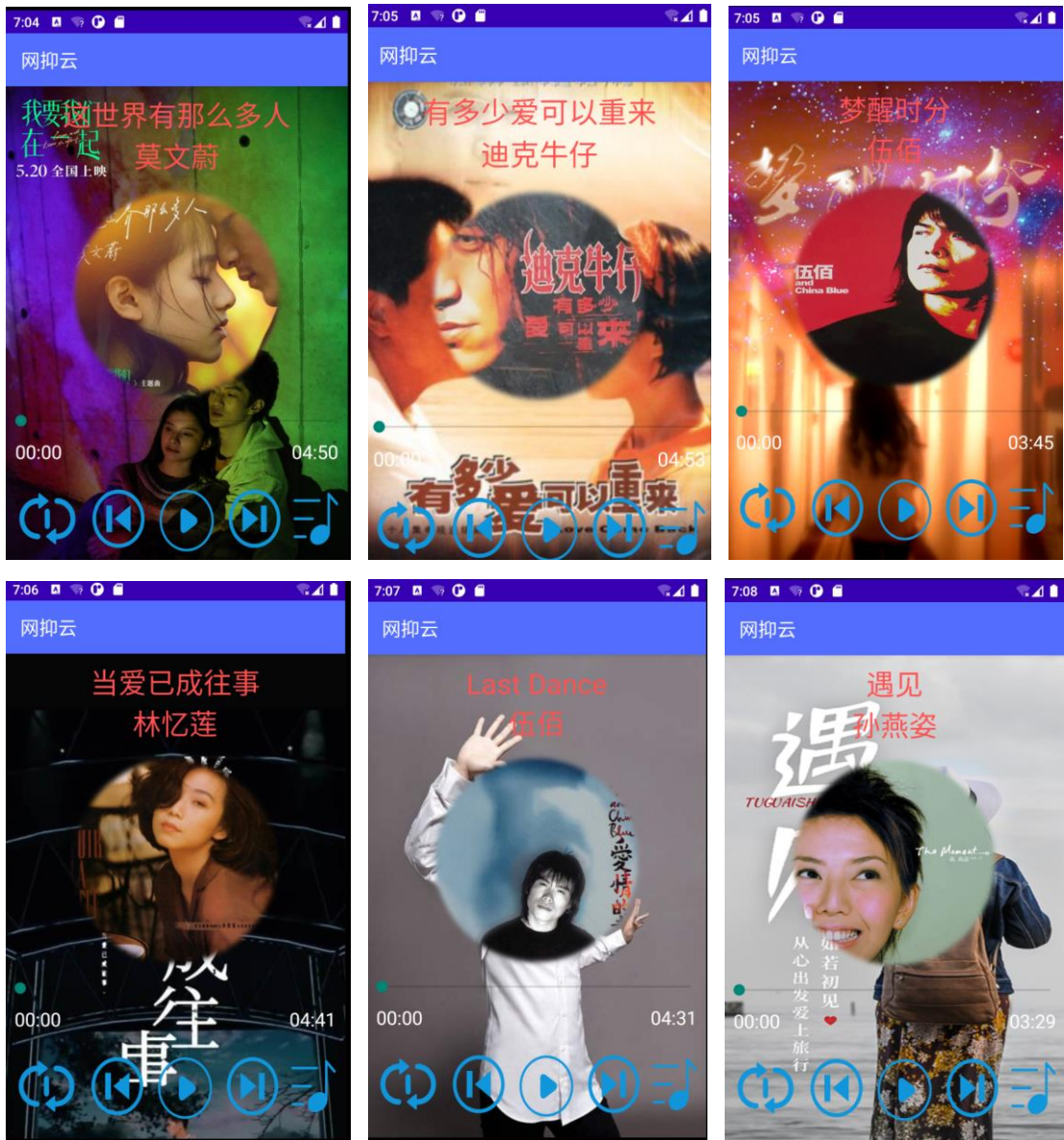


新项目界面：

默认初始界面：



不同歌曲的播放界面：



下面阐述一下界面改进的介绍和实现方式：

1.歌曲列表采用了 RecyclerView 的方式，选择了 6 首歌曲进行垂直排列，每一首歌曲都服从一个模板，模板的样式为左边为歌曲图标，右侧歌曲名称和歌手名字，文字采用垂直对齐，即垂直方向中心对齐，水平方向靠左显示。首先在 drawable 文件件中导入 6 张对应的音乐图片文件。然后制作列表界面，首先在布局文件夹中创建 activity\_music\_list 的 xml 文件，相应格式为，首行为“歌曲列表”，

下面使用 `RecyclerView`，因此要创建相应的模板文件，同样在布局文件夹中创建 `item.xml` 文件，其模板样式为左侧为歌曲图标，右侧为文字，对齐方式为垂直对齐，即 `android:gravity="center_vertical"`。并在主逻辑文件 `Music_list.java` 中写出相应代码，将模板格式联系到布局文件中。所有的相关代码见附录。

2.进入到相应歌曲的播放界面，背景和转盘图片各不相同，上面为歌曲名字和歌手名字，，转盘下面为进度条，下面可以显示总时长和现在时长进度，最下面为功能键，相应功能及实现将在下文中介绍。首先在 `drawable` 文件夹中导入相应的播放歌曲的图片和背景图片，同时导入下面功能键的图片资源。然后在布局文件 `activity_main.xml` 文件中进行排版和设置。然后在主逻辑代码 `MainActivity.java` 中声明相应的控件变量，为了能够在不同歌曲中分别切换不同的转盘图片和背景图片以及标题文字，首先定义了一个 `int` 类型变量 `pos`，用来表示当前是第几个音乐，为了使其能够在不同 `class` 中调用，将其设为 `static`。然后可以利用这个 `pos` 来调用不同的 `Music_list.java` 中的对应的数组相应的位置的变量。因此这个 `pos` 就要等于 `Music_list.java` 中的 `pos`，所以同样地应当将上述的数组和变量都设为 `static` 以便在不同类中互相调用。

除此之外当然也要根据不同的音乐播放不同的音乐资源，音乐资源存放在了 `raw` 这个文件夹里，在 `MusicPlayer.java` 这个 `service` 程序里面，要根据不同的音乐选择不同的资源，也是根据 `pos` 这个变量来选择的。

Music\_list.java 中相应变量如下：

```
public static int[] images = {R.drawable.music1, R.drawable.music2,
R.drawable.music3, R.drawable.music4, R.drawable.music5,
R.drawable.music6};
public static int[] bg = {R.drawable.bg1, R.drawable.bg2, R.drawable.bg3,
R.drawable.bg4, R.drawable.bg5, R.drawable.bg6};
public static String[] names = {"这世界有那么多人\n 莫文蔚", "有多少爱可以重
来\n 迪克牛仔", "梦醒时分\n 伍佰", "当爱已成往事\n 林忆莲", "Last Dance\n 伍佰
", "遇见\n 孙燕姿"};
public static int pos;
```

MainActivity.java 中替换图片和文字的对应代码如下：

```
pos = Music_list.pos;
alayout=(LinearLayout) findViewById(R.id.layout_main);
alayout.setBackgroundResource(Music_list.bg[pos]);
iv_cover = findViewById(R.id.iv_cover);
iv_cover.setImageResource(Music_list.images[pos]);
tv_title = findViewById(R.id.tv_title);
tv_title.setText(Music_list.names[pos]);
```

MusicPlayer.java 中选择不同音乐资源对应代码如下：

```
int position = MainActivity.pos;
switch (position){
    case 0: {
        player = MediaPlayer.create(getApplicationContext(),
R.raw.music1); //加载多媒体文件
        break;
    }
    case 1: {
        player = MediaPlayer.create(getApplicationContext(),
R.raw.music2); //加载多媒体文件
        break;
    }
    case 2: {
        player = MediaPlayer.create(getApplicationContext(),
R.raw.music3); //加载多媒体文件
        break;
    }
    case 3: {
        player = MediaPlayer.create(getApplicationContext(),
R.raw.music4); //加载多媒体文件
        break;
    }
}
```

```

        case 4: {
            player = MediaPlayer.create(getApplicationContext(),
R.raw.music5); //加载多媒体文件
            break;
        }
        case 5: {
            player = MediaPlayer.create(getApplicationContext(),
R.raw.music6); //加载多媒体文件
            break;
        }
    }
}

```

## 1.2 功能介绍与详解


### 1.2.1 歌曲列表与歌曲播放界面的相互跳转

在歌曲列表中点击相应歌曲的栏目就可以跳转进入相应歌曲的播放界面，实现方式如下：在 `Music_list.java` 中添加点击响应，即创建一个意图，实现两界面的跳转，值得注意的是，这里需要将 `position` 赋值给 `pos` 变量，从而使得跳转到 `MainActivity.java` 中知道这是第几个音乐，从而可以将相应的图片和文字设为相应歌曲的。相应代码如下：

```

@Override
public void onClick(View view) {
    Intent mintent = null;
    mintent = new Intent(Music_list.this(getApplicationContext(),
MainActivity.class)
    startActivity(mintent);
    pos = position;
}

```

在歌曲播放界面可以点击  按钮来回到歌曲列表，也就是要在该按钮添加点击响应，相应代码如下：

```

case R.id.btn_list:
    finish();
    break;

```

### 1.2.2 播放或暂停功能



点击此按钮可以暂停音乐的播放。



点击此按钮可以继续播放音乐。

在实现中，以上两个按钮其实为一个按钮，通过点击进行相应的变换，在逻辑上，首先声明一个 `int` 变量 `play`，初始值为 0，每点击按钮一次就加 1，根据 `play` 的奇偶分别进行不同的操作，一个为继续播放，光盘继续转，同时将图案改为暂停图案。另一个为停止播放，光盘停止转，同时将图案该为播放图案。当 `play` 为初始值 0 时与他们不同，要调用 `play()`和 `start()`方法。相关的逻辑代码如下：

```
case R.id.btn_play:
    if (play == 0 && tv_progress.getText().equals("00:00")) {
        // 播放音乐
        control.play();
        // 光盘开始转
        animator.start();
        btn_play.setBackgroundResource(R.drawable.pause);
    }
    play++;
    if (play != 0) {
        if (play % 2 != 0) {
            // 继续播放音乐
            control.continuePlay();
            // 光盘继续转
            animator.resume();
            btn_play.setBackgroundResource(R.drawable.pause);
        } else {
            // 停止播放音乐
            control.pausePlay();
            // 光盘停止转
            animator.pause();
            btn_play.setBackgroundResource(R.drawable.play);
        }
    }
    break;
```

其中的细节问题将在 1.3 中详解。



### 1.2.3 上一首和下一首功能



点击此按钮可以播放上一首歌。



点击此按钮可以播放下一首歌。

实现方式即为给两个按钮添加相应的不同点击响应。两者的大体方法相似，首先点击后需要改变 `pos` 的值，上一首即 `pos` 要减 1，下一首即 `pos` 要加 1。然后就根据新的 `pos` 的值，设置对应的标题和光盘及背景图片，其实现方法与上文类似，然后利用 `init()` 方法重新设置新的歌曲的资源，然后将转盘归位，播放按钮设置为暂停状态时的按钮。

两个按钮对应的代码如下：

```
case R.id.btn_next:
    if(pos < Music_list.names.length-1) {
        pos++;
        tv_title.setText(Music_list.names[pos]);
        iv_cover.setImageResource(Music_list.images[pos]);
        control.play();
        //停止播放音乐
        control.pausePlay();
        //光盘停止转
        animator.pause();
        init();
        animator.start();
        animator.pause();
        btn_play.setBackgroundResource(R.drawable.play);
    }
    break;
case R.id.btn_last:
    if (pos > 0) {
        pos--;
        tv_title.setText(Music_list.names[pos]);
        iv_cover.setImageResource(Music_list.images[pos]);
        control.play();
        //停止播放音乐
```

```

        control.pausePlay();
        //光盘停止转
        animator.pause();
        init();
        animator.start();
        animator.pause();
        btn_play.setBackgroundResource(R.drawable.play);
    }
    break;

```

其中的细节问题将在 1.3 中详解。

### 1.2.4 单曲循环与列表循环的选择

本项目还实现了单曲循环和列表循环的功能作为其中一个拓展功能。



显示此图标，表示当前为单曲循环状态，即播放完当前曲目后会继续播放当前曲目。点击此按钮将会转为列表循环状态。



显示此图标，表示当前为列表循环状态，即播放完当前曲目后会继续播放下一首曲目。点击此按钮将会转化为单曲循环状态。

其实现方式也是添加相应点击响应，以上按钮同样为同一按钮，只是显示图标不同，因此思路与 1.2.3 相似，创建一个 `int` 变量 `circle`，初始值为 0，每点击一次，就将 `circle` 加 1，根据 `circle` 的奇偶改变按钮的显示状态，为偶数时，为单曲循环，为奇数时，为列表循环。相应代码如下：

```

case R.id.btn_circle:
    circle++;
    if(circle%2==0){
        btn_circle.setBackgroundResource(R.drawable.circle1);
    }else{
        btn_circle.setBackgroundResource(R.drawable.circle2);
    }

```

```
}  
break;
```

除了图标显示的切换外，还要添加逻辑代码，找到进度条行进过程的监听代码，对此进行修改，当进度条到头后，首先进行判断，如果 `circle` 为偶数，也就是单曲循环状态，那么就播放完该音乐后，重新调用 `play()` 方法继续从头播放该曲目。如果 `circle` 为奇数，也就是列表循环状态，那么就在播放完该曲目后接着播放下一首曲目，其大体思路与下一首播放功能类似，但有些细节问题将在 1.3 中详解，如果该曲目已经是最后一首，那么在其播完后就将音乐播放停止。其对应的代码如下：

```
if (i== seekBar.getMax() ){  
    if(circle%2==0){  
        //播放音乐  
        control.play();  
        //光盘开始转  
        animator.start();  
        btn_play.setBackgroundResource(R.drawable.pause);  
    }else{  
        if(pos < Music_list.names.length-1) {  
            control.play();  
            //停止播放音乐  
            control.pausePlay();  
            //光盘停止转  
            try  
            {  
                Thread.sleep(2000); //单位：毫秒  
            } catch (Exception e) {  
            }  
            pos++;  
            tv_title.setText(Music_list.names[pos]);  
            iv_cover.setImageResource(Music_list.images[pos]);  
            control.play();  
            //停止播放音乐  
            control.pausePlay();  
            //光盘停止转  
            animator.pause();  
        }  
    }  
}
```

```

        init();
        control.play();
        //光盘开始转
        animator.start();
        btn_play.setBackgroundResource(R.drawable.pause);
        play++;

    }else{
        //停止播放音乐
        control.pausePlay();
        //光盘停止转
        animator.pause();
        btn_play.setBackgroundResource(R.drawable.play);
        play++;
    }
}
}
}

```

### 1.3 细节问题的解决和优化

该项目中存在着很多细节上的问题，在该小节中进行详细阐述。

#### 1.3.1 播放时的细节问题

播放时存在一个细节问题，就是当用户在音乐没有播放过之前拉进度条到某点处时，此时再点播放键，会导致从头播放，因此要对播放时 `play()` 方法的使用加上条件限制，对应的代码如下：

```

if(play==0 && tv_progress.getText().equals("00:00")){
    //播放音乐
    control.play();
    //光盘开始转
    animator.start();
    btn_play.setBackgroundResource(R.drawable.pause);
}

```

也就是当进度条还在初始位置时，并且 `play=0` 时点击播放按钮才调用 `play()` 方法，否则都将调用 `continue()` 方法，这样就可以解决该问题。

#### 1.3.2 上一首和下一首时的细节问题

在使用上一首和下一首时，也会有一定的细节问题，就是会出现

切换到下一首后，进度条和转盘都没有归零，也就是没有回到初始位置，而且上一首的音乐资源也不会释放，因此我们必须在操作之前进行一个归位操作进行重置，在这里我们使用了这样的方法：

```
control.play();
control.pausePlay();
animator.start();
animator.pause();
```

也就是，让音乐控制器开始播放再暂停就可以重置回到初始位置，让转盘也开始转动再暂停，即可以让转盘重置回初始位置，这样切换到上一首和下一首时都可以从初始位置开始播放，也可以将原有资源进行重置和释放。

### 1.3.3 列表循环时的细节问题

在进行列表循环时会遇到一个比较棘手的问题，那就是由于进度条归位比较慢，往往在进度条到尽头后，切换到下一首后，进度条还没有完全归位，但此时进度条监听的函数是时时刻刻运行的，所以切换到下一首后，还会监听到进度条此时在最大位置处，所以会再次运行一次播完切换到下一首的程序，所以就会出现一次切换两首的现象。经过思考，可以在合适的位置加入延时程序，让程序等待进度条完全归位后在运行，这样就可以避免上述现象的发生，经过多次调试，最终找到了合适的位置并找到了合适的延时时间。相关代码如下（高亮部分即延时程序）：

```
if(pos < Music_list.names.length-1) {
    control.play();
    //停止播放音乐
    control.pausePlay();
    //光盘停止转
    delay(1000);
}
```

```

    }
    Thread.sleep(2000); //单位: 毫秒
    } catch (Exception e) {
    }
    pos++;

```

### 1.3.4 进度条调整时的细节问题

在调试程序时，发现进度条的设置存在一些问题，首先在没有播放音乐时，如果拉动进度条到某一位置，这时放开是不会播放的，所以第一次移动进度条，必须调用 `play()` 方法，进行启动程序，也就是 `play` 等于 0 时，调用 `play()` 方法。

```

if(play == 0) {
    control.play();
    animator.start();
    control.pausePlay();
    animator.pause();
}

```

另一个问题就是，如果此时已经为暂停状态，再拖动进度条时，也会继续播放，这是不符合要求的，所以要在释放进度条时进行检查是否在暂停状态，如果不在暂停状态就继续播放，如果在暂停状态则不再播放。相关代码如下：

```

@Override
//用户停止滑动进度条的监听
public void onStopTrackingTouch(SeekBar seekBar) {
    if(play%2!=0){
        control.continuePlay();
        animator.resume();
    }
}
}

```

## 2. 附录

### 1) 布局文件代码

#### 1. 音乐列表主布局（activity\_music\_list.xml）代码：

```
<?xml version="1.0" encoding="utf-8" ?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    tools:context=".Music_list"

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="歌曲列表"
        android:gravity="center"
        android:textSize="35sp"
        android:layout_gravity="center"
        android:background="#EC6565"/>

        <androidx.recyclerview.widget.RecyclerView
            android:id="@+id/rv"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:background="#FBF6F9"/>
    </LinearLayout>
```

#### 2. 音乐列表布局对应的模板布局（item.xml）代码

```
<?xml version="1.0" encoding="utf-8" ?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"

    <ImageView
        android:id="@+id/iv_image"
        android:layout_width="150dp"
```

```

        android:layout_height="150dp"
        android:scaleType="fitXY"/>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        >
        <TextView
            android:id="@+id/tv_name"
            android:layout_width="260dp"
            android:layout_height="150dp"
            android:background="#BFBBBB"
            android:padding="10dp"
            android:text="歌曲名"
            android:gravity="center_vertical"
            android:textColor="#070607"
            android:textSize="20sp" />

    </LinearLayout>
</LinearLayout>

```

### 3. 歌曲播放界面布局（activity\_main.xml）代码

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity"
    android:background="@drawable/bg"
    android:orientation="vertical"
    android:id="@+id/layout_main">
    <TextView
        android:id="@+id/tv_title"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="这世界有那么多人"
        android:gravity="center"
        android:layout_marginTop="10dp"
        android:textColor="#FF5252"
        android:textSize="30sp"/>

    <ImageView
        android:id="@+id/iv_cover"

```



```
        android:layout_width="230dp"
        android:layout_height="230dp"
        android:layout_gravity="center_horizontal"
        android:layout_marginTop="10dp"
        android:src="@drawable/music1" />
```

```
<SeekBar
```

```
    android:id="@+id/sb"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="10dp"/>
```

```
<RelativeLayout
```

```
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="10dp">
```

```
    <TextView
```

```
        android:id="@+id/tv_progress"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="00:00"
        android:textColor="@color/white"
        android:textSize="20sp" />
```

```
    <TextView
```

```
        android:id="@+id/tv_total"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="04:50"
        android:layout_alignParentRight="true"
        android:textSize="20sp"
        android:textColor="@color/white"/>
```

```
<!--          沿着母版的右边排列——排列属性-->
```

```
</RelativeLayout>
```

```
<LinearLayout
```

```
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:layout_margin="5dp">
```

```
    <Button
```

```
        android:id="@+id/btn_circle"
        android:layout_width="75dp"
        android:layout_height="75dp"
```

```

        android:layout_weight="1"
        android:layout_margin="5dp"
        android:background="@drawable/circle1"

    />
    <Button
        android:id="@+id/btn_last"
        android:layout_width="70dp"
        android:layout_height="70dp"
        android:layout_weight="1"
        android:layout_margin="5dp"
        android:background="@drawable/last"

    />
    <Button
        android:id="@+id/btn_play"
        android:layout_width="65dp"
        android:layout_height="65dp"
        android:layout_weight="1"
        android:layout_margin="7dp"
        android:background="@drawable/play"

    />
    <Button
        android:id="@+id/btn_next"
        android:layout_width="70dp"
        android:layout_height="70dp"
        android:layout_weight="1"
        android:layout_margin="5dp"
        android:background="@drawable/next"

    <Button
        android:id="@+id/btn_list"
        android:layout_width="55dp"
        android:layout_height="55dp"
        android:layout_margin="5dp"
        android:background="@drawable/list"
        android:layout_weight="1"

    </LinearLayout>

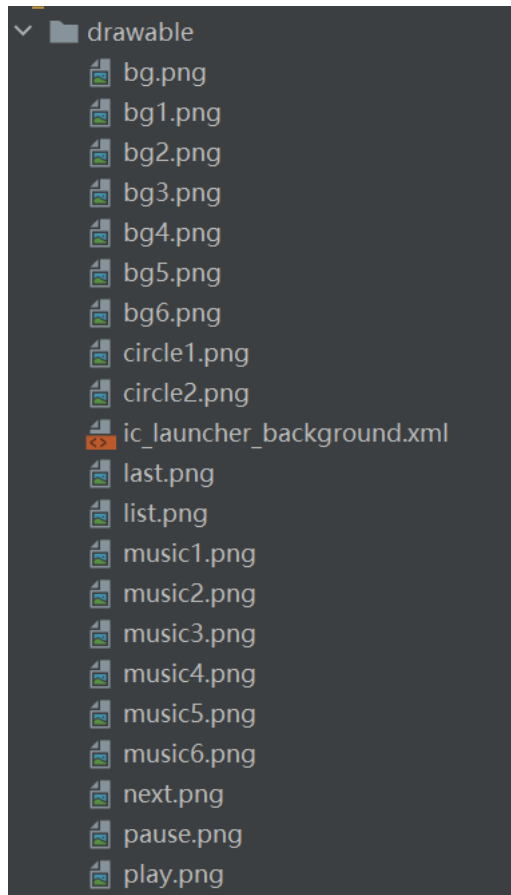
</LinearLayout>

```

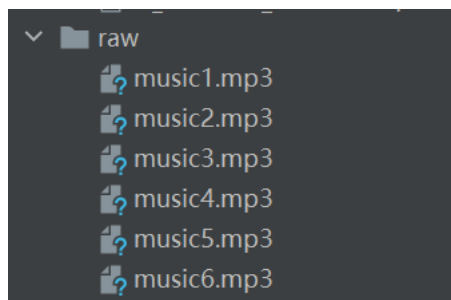
## 2) 资源文件

### 1. drawable 文件夹中为所有转盘图片和背景图片以及功能键图标的

图片资源。



## 2. raw 文件夹中为音乐资源资料



## 2) 界面逻辑程序代码

### 1.. 音乐列表界面逻辑程序 (Music\_list.java) 代码

```
package com.example.playerapp;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;
```

```

import android.content.Intent;
import android.graphics.Rect;
import android.os.Bundle;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;

public class Music_list extends AppCompatActivity {
    private RecyclerView mrv;

    public static int[] images = {R.drawable.music1, R.drawable.music2,
R.drawable.music3, R.drawable.music4, R.drawable.music5,
R.drawable.music6};

    public static int[] bg = {R.drawable.bg1, R.drawable.bg2,
R.drawable.bg3, R.drawable.bg4, R.drawable.bg5, R.drawable.bg6};

    public static String[] names = {"这世界有那么多人\n 莫文蔚", "有多少爱可
以重来\n 迪克牛仔", "梦醒时分\n 伍佰", "当爱已成往事\n 林忆莲", "Last Dance\n
伍佰", "遇见\n 孙燕姿"};

    public static int pos;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_music_list);
        mrv = findViewById(R.id.rv);
        mrv.addItemDecoration(new MyDecoration());
        mrv.setLayoutManager(new LinearLayoutManager(Music_list.this,
RecyclerView.VERTICAL, false));
        mrv.setAdapter(new MyAdapter());
    }

    private class MyAdapter extends RecyclerView.Adapter<MyHolder> {
        @NonNull
        @Override
        public MyHolder onCreateViewHolder(@NonNull ViewGroup parent, int
viewType) {
            View myView = View.inflate(getApplicationContext(),
R.layout.item, null);
            MyHolder myHolder = new MyHolder(myView);

            // 或者 MainActivity.class
            return myHolder;
        }
    }

```

```

        @Override
        public void onBindViewHolder(@NonNull MyHolder holder, int
position) {
            holder.miv_image.setBackgroundResource(images[position]);
            holder.mtv_name.setText(names[position]);
            holder.itemView.setOnClickListener(new
View.OnClickListener() {
                @Override
                public void onClick(View view) {
                    Intent mintent = null;
                    mintent = new
Intent(Music_list.this.getApplicationContext(), MainActivity.class);
                    startActivity(mintent);
                    pos = position;
                }
            });
        }

        @Override
        public int getItemCount() {
            return names.length;
        }
    }

    private class MyHolder extends RecyclerView.ViewHolder {
        ImageView miv_image;
        TextView mtv_name;

        public MyHolder(@NonNull View itemView) {
            super(itemView);
            miv_image = itemView.findViewById(R.id.iv_image);
            mtv_name = itemView.findViewById(R.id.tv_name);
        }
    }

    private class MyDecoration extends RecyclerView.ItemDecoration {
        @Override
        public void getItemOffsets(@NonNull Rect outRect, @NonNull View
view, @NonNull RecyclerView parent, @NonNull RecyclerView.State state)
{

```

```

        super.getItemOffsets(outRect, view, parent, state);
        outRect.set(0, 0, 0, 20);
    }
}
}

```

## 2. 音乐播放界面逻辑程序（MainActivity.java）代码

```

package com.example.playerapp;

import static java.lang.Integer.parseInt;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import android.animation.ObjectAnimator;
import android.content.ComponentName;
import android.content.Intent;
import android.content.ServiceConnection;
import android.graphics.Typeface;
import android.media.MediaPlayer;
import android.os.Bundle;
import android.os.Handler;
import android.os.HandlerThread;
import android.os.IBinder;
import android.os.Looper;
import android.os.Message;
import android.view.View;
import android.view.animation.LinearInterpolator;
import android.widget.Button;
import android.widget.ImageButton;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.SeekBar;
import android.widget.TextView;

import java.util.Timer;

public class MainActivity extends AppCompatActivity{
    private ImageView iv_cover;
    private static SeekBar sb;
    private static TextView tv_progress,tv_total,tv_title;
    private Button btn_next,btn_last,btn_list;
    private Button btn_play,btn_circle;
    public static int pos;

```

```

int play = 0;int circle = 0;
public static int duration;
LinearLayout alayout;

private ObjectAnimator animator; //声明一个动画组件 ObjectAnimator

private MediaPlayer.MusicControl control;//声明 MusicService 中的音乐
控制器

private ServiceConnection connection = new ServiceConnection() { //
声明服务连接
    @Override
    public void onServiceConnected(ComponentName componentName,
IBinder iBinder) {
        control = (MediaPlayer.MusicControl) iBinder;//实例化音乐控制
对象，即 control。
    }
    @Override
    public void onServiceDisconnected(ComponentName componentName) {

    }
};

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    pos = Music_list.pos;
    init();
}
public void init(){
    play = 0;
    alayout=(LinearLayout) findViewById(R.id.layout_main);
    alayout.setBackgroundResource(Music_list.bg(pos));
    iv_cover = findViewById(R.id.iv_cover);
    iv_cover.setImageResource(Music_list.images(pos));
    tv_title = findViewById(R.id.tv_title);
    tv_title.setText(Music_list.names(pos));
    sb = findViewById(R.id.sb);
    tv_progress = findViewById(R.id.tv_progress);
    tv_total = findViewById(R.id.tv_total);
    // tv_progress.setTypeface(Typeface.createFromAsset(getAssets(),
"newfont.ttf"));
}

```

```
// tv_total.setTypeface (Typeface.createFromAsset (getAssets (),
"newfont.ttf"));

    btn_play = findViewById(R.id.btn_play);
    btn_next = findViewById(R.id.btn_next);
    btn_last = findViewById(R.id.btn_last);
    btn_list = findViewById(R.id.btn_list);
    btn_circle = findViewById(R.id.btn_circle);

    OnClickListener monclick = new OnClickListener();
    btn_play.setOnClickListener(monclick);
    btn_next.setOnClickListener(monclick);
    btn_last.setOnClickListener(monclick);
    btn_list.setOnClickListener(monclick);
    btn_circle.setOnClickListener(monclick);

    //执行动画的对象是 iv_cover, // 动画效果是 0-360°旋转（用的是浮点数，所以
    加个f）。
    animator =
ObjectAnimator.ofFloat(iv_cover,"rotation",0.0f,360.0f);
    animator.setDuration(10000); //旋转一周的时长，单位是毫秒，此处设置了
    10s
    animator.setInterpolator(new LinearInterpolator()); //设置匀速转动
    animator.setRepeatCount(-1); //设置循环，此处设置的是无限循环。如果是正
    值，意味着转动多少圈。

    //声明一个意图，该意图进行服务的启动，意思是将 MusicService 里面的服务要传
    到主程序这里来。
    Intent mintent = new Intent(MainActivity.this,MusicPlayer.class);
    bindService(mintent,connection,BIND_AUTO_CREATE); //建立意图中
    MainActivity 与 MusicService 两对象的服务连接

    seekBarListener msbListener = new seekBarListener();
    sb.setOnSeekBarChangeListener(msbListener);

}
// 设置播放、暂停、继续和退出按钮的监听（或点击）事件
class OnClickListener implements View.OnClickListener{

    @Override
    public void onClick(View view) {
        switch (view.getId()){
            case R.id.btn_play:
```



```

        if(play==0){
tv_progress.getText().equals("00:00")){
        //播放音乐
        control.play();
        //光盘开始转
        animator.start();
        btn_play.setBackgroundResource(R.drawable.pause);
    }
    play++;
    if(play!=0){
        if (play % 2 != 0) {
            //继续播放音乐
            control.continuePlay();
            //光盘继续转
            animator.resume();
        }
        btn_play.setBackgroundResource(R.drawable.pause);
    } else {
        //停止播放音乐
        control.pausePlay();
        //光盘停止转
        animator.pause();
        btn_play.setBackgroundResource(R.drawable.play);
    }
    break;
    case R.id.btn_next:
        if(pos < Music_list.names.length-1){
            pos++;
            tv_title.setText(Music_list.names[pos]);
            iv_cover.setImageResource(Music_list.images[pos]);
            control.play();
            //停止播放音乐
            control.pausePlay();
            //光盘停止转
            animator.pause();
            init();
            animator.start();
            animator.pause();
            btn_play.setBackgroundResource(R.drawable.play);
        }
        break;
    case R.id.btn_last:

```

```

        if (pos > 0) {
            pos--;
            tv_title.setText(Music_list.names[pos]);
            iv_cover.setImageResource(Music_list.images[pos]);
            control.play();
            //停止播放音乐
            control.pausePlay();
            //光盘停止转
            animator.pause();
            init();
            animator.start();
            animator.pause();
            btn_play.setBackgroundResource(R.drawable.play);
        }
        break;
        case R.id.btn_list:
            finish();
            break;
        case R.id.btn_circle:
            circle++;
            if(circle%2==0){
                btn_circle.setBackgroundResource(R.drawable.circle1);
            }else{
                btn_circle.setBackgroundResource(R.drawable.circle2);
            }
            break;
    }
}

@Override
protected void onDestroy() {
    control.stopPlay();
    unbindService(connection);
    super.onDestroy();
}

//Handler 主要用于异步消息的处理，在这里是处理子线程 MusicService 传来的消息
public static Handler handler = new Handler(Looper.getMainLooper()){

    @Override

```

```
public void handleMessage(@NonNull Message msg) {
    //super.handleMessage(msg);
    Bundle bundle = msg.getData();
    duration = bundle.getInt("duration");//把音乐时长放在 bundle 里
    int currentDuration = bundle.getInt("currentDuration");//把
    音乐当前播放时长放在 bundle 里

    sb.setMax(duration);
    sb.setProgress(currentDuration);

    //显示总时长
    int minite = duration / 1000 / 60;
    int second = duration / 1000 % 60;
    String strMinite = "";
    String strSecond = "";
    if (minite < 10){
        strMinite = "0" +minite;
    }else {
        strMinite = minite + "";
    }
    if (second < 10){
        strSecond = "0" + second;
    }else {
        strSecond = second + "";
    }
    tv_total.setText(strMinite + ":" + strSecond);

    //显示播放时长
    minite = currentDuration / 1000 / 60;
    second = currentDuration / 1000 % 60;

    if (minite < 10){
        strMinite = "0" +minite;
    }else {
        strMinite = minite + "";
    }
    if (second < 10){
        strSecond = "0" + second;
    }else {
        strSecond = second + "";
    }
    tv_progress.setText(strMinite + ":" + strSecond);
}
```

```

};

//给进度条设置监听
class seekBarListener implements SeekBar.OnSeekBarChangeListener {
    @Override
    //进度条行进过程的监听
    public void onProgressChanged(SeekBar seekBar, int i, boolean b)
{
    if (i== seekBar.getMax()) {
        if(circle%2==0){
            //播放音乐
            control.play();
            //光盘开始转
            animator.start();
            btn_play.setBackgroundResource(R.drawable.pause);
        }else{
            if(pos < Music_list.names.length-1){
                control.play();
                //停止播放音乐
                control.pausePlay();
                //光盘停止转
                try
                {
                    Thread.sleep(2000); //单位：毫秒
                } catch (Exception e) {}
            }
            pos++;
            tv_title.setText(Music_list.names[pos]);
            iv_cover.setImageResource(Music_list.images[pos]);
            control.play();
            //停止播放音乐
            control.pausePlay();
            //光盘停止转
            animator.pause();

            init();
            control.play();
            //光盘开始转
            animator.start();
            btn_play.setBackgroundResource(R.drawable.pause);
            play++;
        }else{
            //停止播放音乐

```



```
import android.content.Intent;
import android.media.MediaPlayer;
import android.os.Binder;
import android.os.Build;
import android.os.Bundle;
import android.os.IBinder;
import android.os.Message;

import androidx.annotation.Nullable;

import java.util.Timer;
import java.util.TimerTask;

/**
 * 在这里设置音乐播放功能的服务
 *
 */

public class MusicPlayer extends Service {

    // 设置两个成员变量
    private MediaPlayer player; // 声明一个多媒体对象
    private Timer timer; // 声明一个时钟对象

    public MusicPlayer () {}

    @Nullable
    @Override
    public IBinder onBind(Intent intent) {
        return new MusicControl(); // 将 MusicControl() 返回给 onBind() 方法，
        // 这样绑定服务的时候，可以把音乐控制器实例化。
    }

    @Override
    public void onCreate() {
        super.onCreate();
        player = new MediaPlayer(); // 实例化多媒体
    }

    @Override
    public void onDestroy() {
        super.onDestroy();
        if (player == null) return;
        // if (player.isLooping()) player.stop(); // 停止播放音乐
    }
}
```

```

        player.release();//释放资源
        player = null;
    }

```

//创建一个内部类 MusicControl，功能是让主程序控制 service 里面的多媒体对象。  
IBinder 是 Binder 的子类，因此要返回 MusicControl 给 IBinder。

```

    class MusicControl extends Binder{
        public void reset(){
            try{
                player.reset();//重置音乐播放器
            }catch (Exception exception) { //catch 用来处理播放时产生的异常
                exception.printStackTrace();
            }
        }
        public void play() { //开始播放
            try{
                player.reset();//重置音乐播放器
                int position = MainActivity.pos;
                switch (position){
                    case 0: {
                        player =
MediaPlayer.create(getApplicationContext(), R.raw.music1); //加载多媒体
文件
                        break;
                    }
                    case 1: {
                        player =
MediaPlayer.create(getApplicationContext(), R.raw.music2); //加载多媒体
文件
                        break;
                    }
                    case 2: {
                        player =
MediaPlayer.create(getApplicationContext(), R.raw.music3); //加载多媒体
文件
                        break;
                    }
                    case 3: {
                        player =
MediaPlayer.create(getApplicationContext(), R.raw.music4); //加载多媒体
文件
                        break;
                    }
                    case 4: {
                        player =

```

```

MediaPlayer.create(getApplicationContext(), R.raw.music5); //加载多媒体
文件
                break;
            }
            case 5: {
                player =
MediaPlayer.create(getApplicationContext(), R.raw.music6); //加载多媒体
文件
                break;
            }
        }
        player.start(); //开始播放音乐
        addTimer(); //添加计时器
    } catch (Exception exception) { //catch 用来处理播放时产生的异常
        exception.printStackTrace();
    }
}

public void pausePlay() { //暂停播放
    player.pause();
}

public void continuePlay() { //继续播放
    player.start();
}

public void stopPlay() { //暂停播放
    player.stop();
    player.release();
    try {
        timer.cancel();
    } catch (Exception e) {
        e.printStackTrace();
    }
}

public void seekTo(int progress) { //定义播放位置播放
    player.seekTo(progress);
}
}

//添加计时器，计时器是一个多线程的东西，用于设置音乐播放器中的进度条信息
public void addTimer() {
    if (timer == null) {
        timer = new Timer();
        TimerTask task = new TimerTask() { //实例化一个计时任务对象
            @Override
            public void run() { //run 就是多线程的一个东西，用于 service 和

```



主线程（即 MainActivity）之间的通信

`if (player == null) return;` //如果 player 没有实例化，就不执行下面的代码。

`int duration = player.getDuration();` //获取歌曲总长度

`int currentDuration = player.getCurrentPosition();` //

获取歌曲当前播放进度

//将音乐的总时长、播放时长封装到消息对象中去；

`Message message =`

`MainActivity.handler.obtainMessage();` //在主线程获取一个消息空间

`Bundle bundle = new Bundle();` //定义一个包裹，将歌曲总长度和当前播放长度打包放进去

`bundle.putInt("duration",duration);`

`bundle.putInt("currentDuration",currentDuration);`

`message.setData(bundle);` //将消息包括给 message

`MainActivity.handler.sendMessage(message);` //将消息添加到主线程中

`}`

`};`

//开始计时任务后 5ms，执行第一次任务，以后每 500ms 执行一次任务

`timer.schedule(task, 5,1000);`

`}`

`}`

`}`