中山大学移动信息工程学院本科生实验报告

（2017 年秋季学期）

课程名称： 移动应用开发　　**任课教师：** 郑贵锋

年级	15M1	专业（方向）	移动互联网
学号	15352008	姓名	蔡荣裕
电话	13727021990	Email	897389207@qq.com
开始日期	2017/10/18	完成日期	2017/10/22

一、 实验题目

服务与多线程--简单音乐播放器

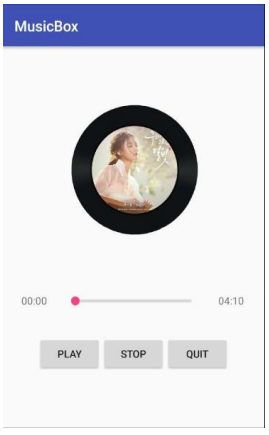
二、 实验目的

1. 学会使用 MediaPlayer
2. 学会简单的多线程编程，使用 Handle 更新 UI
3. 学会使用 Service 进行后台工作
4. 学会使用 Service 与 Activity 进行通信

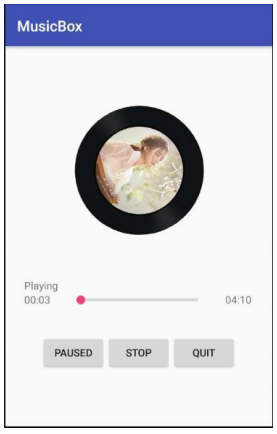
三、 实验内容

实现一个简单的播放器，要求功能有：

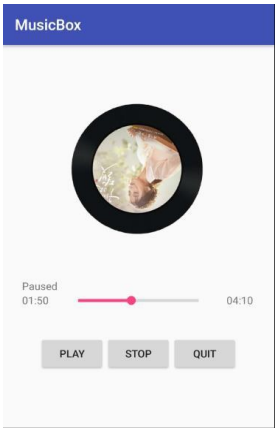
1. 播放、暂停，停止，退出功能。
2. 后台播放功能。
3. 进度条显示播放进度、拖动进度条改变进度功能。
4. 播放时图片旋转，显示当前播放时间功能。



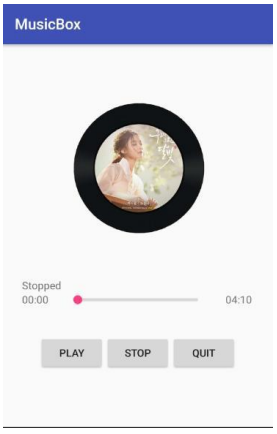
打开程序主页面



开始播放



暂停

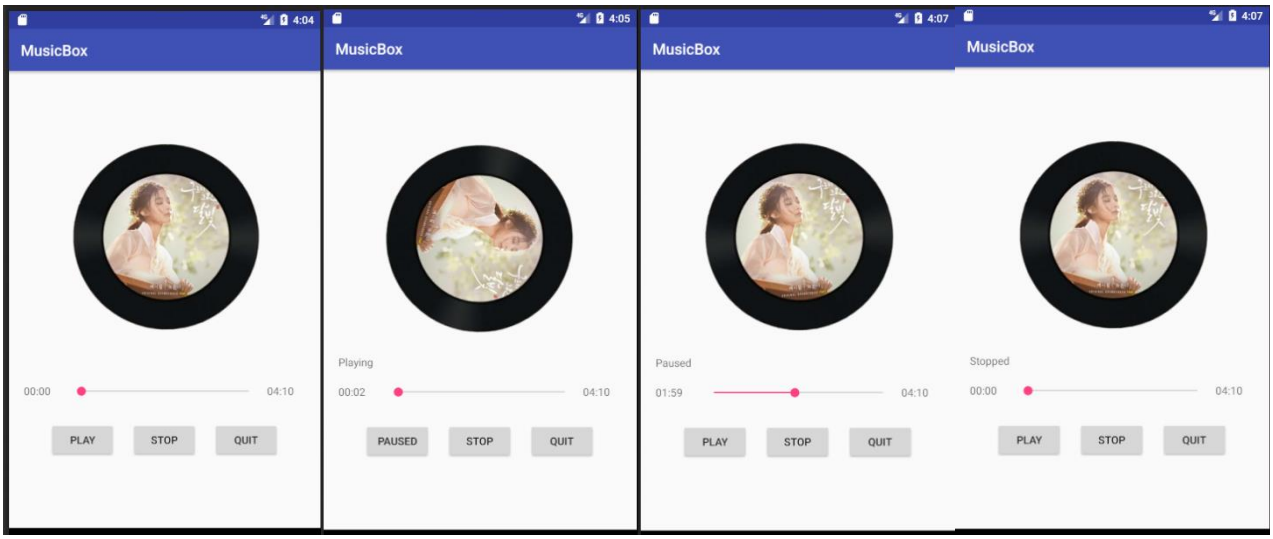


停止

四、 课堂实验结果

（一） 实验截图：

1. 虚拟机上效果如下：



打开程序主页面　　开始播放　　暂停　　停止

（二） 关键步骤：

1. 音乐功能及 Service 的实现

i. 将音乐文件添加到内置 SD 卡中

使用 Tool 中的 Android Device Monitor 将音乐文件添加到内置 SD 卡的音乐目录中，此处需要动用 SDK Tool 中的 adb 进行 root 获取写入虚拟机文件的权限。

ii. 将音乐文件挂载到 MediaPlayer，同时实现播放暂停，停止的接口。

- 获取 SD 卡中音乐文件路径，同时初始化 MediaPlayer。

```
/**
 * 初始化音乐播放器
 */
private void MediaPlayer_Init(){//获取文件路径
    PATHS=Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_MUSIC).toString();
    try {//初始化
        mediaPlayer.setDataSource(PATHS);
        mediaPlayer.prepare();
        mediaPlayer.seekTo(0);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

- 实现音乐播放和暂停接口

```
/**
 * 改变播放状态 暂停和播放互相转换
 */
public void Start_Pause() {
    if (mediaPlayer.isPlaying()) {
        mediaPlayer.pause();
    } else {
        mediaPlayer.start();
    }
}
```

- 音乐停止接口

```
/**
 * 停止当前播放，重新使得 mediaPlayer 回到载入文件之后的初始状态
 */
public void Stop_Play() {
    if (mediaPlayer != null) {
```

```
        mediaPlayer.stop();
        try {
            mediaPlayer.prepare();
            mediaPlayer.seekTo(0);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

iii. 使用 Server 实现 MediaPlayer 播放音乐功能

- 在 Manfast.xml 中注册 Service

```
<service android:name=".MusicService" android:enabled="true" android:exported="true"/>
```

- 实现 Binder 类，通过 Binder 来保持 Activity 和 Service 的通信，通过传入的 code 来判定需要进行的操作，同时通过 Parcel 进行数据的传输，获取输入的数据，并将需要的数据传输回去。此处部分操作直接调用了之前实现的一些函数。

```
protected class MyBinder extends Binder {
    @Override
    protected boolean onTransact(int code, Parcel data, Parcel reply, int flags) throws RemoteException {
        switch (code) {
            case 101:
                Start_Pause(); //播放 or 暂停
                break;
            case 102:
                Stop_Play();//停止
                break;
            case 103:
                mediaPlayer.release();//退出
                break;
            case 104:
                reply.writeInt(mediaPlayer.getCurrentPosition());//更新进度条后调整音乐
                break;
            case 105:
                mediaPlayer.seekTo(data.readInt());//滑动进度条更新播放位置
                break;
            default: {//其他 code 设置 为初始化并回传数据
                MediaPlayer_Init();
                reply.writeInt(mediaPlayer.getDuration());
                reply.writeStringList(PATHS);
                break;
            }
        }
        return super.onTransact(code, data, reply, flags);
    }
}
```

- 在 Activity 实现，建立 Activity 和 Service 的连接的函数，此处主要就是两个需要重载的函数 onServiceConnected 和 onServiceDisconnected，此处 onServiceConnected 的时候也就是连接建立的时候进行歌曲的初始化，同时将初始化返回的数据用于设置 UI，onServiceDisconnected 只需要 IBinder=null 即可。然后使用 startService 和 bindService 建立连接。

```
/**
 * 建立 Activity 和 service 的连接
 */
private void Init_Connect(){
    serviceConnection = new ServiceConnection() {
        @Override
        public void onServiceConnected(ComponentName name, IBinder service) {//连接的建立
            myBinder = (MusicService.MyBinder) service;
            try {
                Parcel data=Parcel.obtain();
                Parcel reply=Parcel.obtain();
                myBinder.transact(106,data,reply,0);//初始化
                int len=reply.readInt();//返回时长
                seekBar.setMax(len); // 设置最大时长
                end.setText(time.format(len));//设置时间
            }catch (Exception e) {
                e.printStackTrace();
            }
        }
        @Override
```

```
        public void onServiceDisconnected(ComponentName name) {//断开连接
            myBinder = null;
        }//连接断开
    };

    Intent intent = new Intent(this, MusicService.class);
    startService(intent);
    bindService(intent, serviceConnection, Context.BIND_AUTO_CREATE);
}
```

iv. 通过 Handler 进行 UI 的更新

- 播放时图片转动此处设定转动时间为 10s，以及点击事件初始化，点击事件改变 sta(相当于 flag)用于告诉 Handler 什么按钮按下了

```
/**
 * 图片转动设置
 */
private void Init_animator(){
    animator = ObjectAnimator.ofFloat(imageView, "rotation", 0.0f, 360.0f);
    //第一个参数是控件，第二个是变化方式，第三个是可变长参数，第 4 个是变化角度
    animator.setDuration(10000); //设定转一圈的时间
    animator.setInterpolator(new LinearInterpolator());//定义动画的变化速率，这里是线性
    animator.setRepeatCount(Animation.INFINITE); //设定无限循环
}
/**
 * 点击事件初始化 sta 为不同点击事件对应 flag，其为 Handle 更新 UI 的依据
 */
private void Init_listener(){
    seekBar.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
        @Override //拖动监听
        public void onProgressChanged(SeekBar seekBar, int i, boolean b) {//拖动中
            if(b){
                try {
                    Parcel data = Parcel.obtain();
                    Parcel reply = Parcel.obtain();
                    data.writeInt(i);
                    myBinder.transact(105, data, reply, 0);//更新播放位置
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        }
        @Override
        public void onStartTrackingTouch(SeekBar seekBar) {}//开始拖动
        @Override
        public void onStopTrackingTouch(SeekBar seekBar) {}//拖动完毕
    });
    play.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {//播放暂停切换
            if(status.getText().toString().equals("Playing"))sta=2;
            else sta=1;
        }
    });
    stop.setOnClickListener(new View.OnClickListener() {//停止
        @Override
        public void onClick(View view) {
            sta=3;
        }
    });//停止
    quit.setOnClickListener(new View.OnClickListener() {//退出
        @Override
        public void onClick(View view) {
            sta=4;
        }
    });//退出
}
```

- 使用 Handler 更新 UI，此处重载函数 handleMessage 来获得传进来的 sta 值，然后根据 sta 值进行对应的 UI 的更新。

```
/**
 * 使用 Handler 更新 UI
 */
private void Init_Handler(){
    final Handler mHandler = new Handler() {
```

```
        @Override
        public void handleMessage(Message msg) {
            super.handleMessage(msg);
            if (msg.what == 1 || msg.what == 2) {//播放 or 暂停
                if (msg.what == 1) {//播放
                    play.setText("Paused");
                    status.setText("Playing");
                    if (animator.isPaused()) animator.resume();
                    else animator.start();
                } else {//暂停
                    play.setText("PLAY");
                    status.setText("Paused");
                    animator.pause();
                }
                try {
                    Parcel data = Parcel.obtain();
                    Parcel reply = Parcel.obtain();
                    myBinder.transact(101, data, reply, 1);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            } else if (msg.what == 3) {//停止
                seekBar.setProgress(0);
                play.setText("PLAY");
                status.setText("Stopped");
                animator.setCurrentPlayTime(0);
                animator.cancel();
                try {
                    Parcel data = Parcel.obtain();
                    Parcel reply = Parcel.obtain();
                    myBinder.transact(102, data, reply, 0);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            } else if (msg.what == 4) {//退出
                try {
                    Parcel data = Parcel.obtain();
                    Parcel reply = Parcel.obtain();
                    myBinder.transact(103, data, reply, 0);
                } catch (Exception e) {
                    e.printStackTrace();
                }
                unbindService(serviceConnection);//停止服务时解除绑定
                serviceConnection = null;
                try {
                    MainActivity.this.finish();
                    System.exit(0);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
            sta = 0; //状态置 0
            try {//音乐进度更新
                Parcel data = Parcel.obtain();
                Parcel reply = Parcel.obtain();
                myBinder.transact(104, data, reply, 0);
                int progress = reply.readInt();
                seekBar.setProgress(progress);
                begin.setText(time.format(progress));
            } catch (Exception e) {
                e.printStackTrace();
            }

        }
    };
    /**
     * 新建子线程定时调用 Hander
     */
    Thread mThread = new Thread(){ //子线程更新
        @Override
        public void run(){
            while (true) {
                try {
                    Thread.sleep(100);
                } catch (Exception e) {
                    e.printStackTrace();
```

```
                }
                mHandler.obtainMessage(sta).sendToTarget();
            }
        }
    };
    mThread.start();
}
```

（三） 实验中遇到的困难和解决思路：

1. 一开始文件无法传进虚拟机，一直提示没有权限：

```
- ddms] transfer error: couldn't create file: Permission denied
 Failed to push selection: couldn't create file: Permission denied
```

解决办法使用 sdk 中的 platform-tools 工具中的 adb 进行强行 root，使用命令 adb root 之后权限问题解决。

```
D:\Android\sdk\platform-tools>adb root
```

2. 音乐文件放在/data 文件夹中无法被获取。
解决方式，直接改放到内置 SD 卡的音乐目录下直接使用 Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_MUSIC)获取。
3. 在主界面选择 back 之后在进入图片不转。
解决办法有 2，第一种再次进入之后判断图片是不是在转，不转的话使用 handler 更新使其重头开始转，缺点每次再进入都是从头转。第二种主界面在按下 back 之后变成后台运行，此时进入时其还在按照时间转动。

```
/**
 * 实现后台
 */
@Override
public void onBackPressed() {
    moveTaskToBack(false);
}
```

五、 课后实验结果

1. 动态权限获取。
虚拟机在使用的时候如果需要读取文件需要有文件的读写权限，可以直接在 Manifast 中注册权限，然后进行动态获取。

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
```

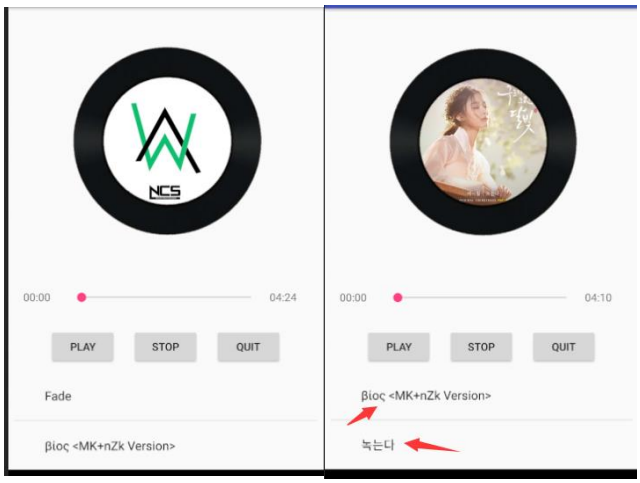
在主函数中实现动态获取

```
/**
 * 动态获取访问 SD 卡权限
 */
private static final int REQUEST_EXTERNAL_STORAGE = 1;
private static String[] PERMISSIONS_STORAGE = {
        Manifest.permission.READ_EXTERNAL_STORAGE,
        Manifest.permission.WRITE_EXTERNAL_STORAGE };
public static void verifyStoragePermissions(Activity activity) {
    int permission = ActivityCompat.checkSelfPermission(activity,
            Manifest.permission.WRITE_EXTERNAL_STORAGE);//直接检测是否否写的权限
    if (permission != PackageManager.PERMISSION_GRANTED) { //没有就去申请
        ActivityCompat.requestPermissions(activity, PERMISSIONS_STORAGE,
                REQUEST_EXTERNAL_STORAGE);
    }
    else PERMISSIONS=true;
}

/**
 * 根据权限返回的结果处理
 */
@Override
```

```
public void onRequestPermissionsResult(int requestCode , String permissions[],int[] grantResults){
    if(grantResults.length>0&&grantResults[0]==PackageManager.PERMISSION_GRANTED){
    }
    else{
        Toast.makeText(MainActivity.this,"没有权限程序退出运行",Toast.LENGTH_SHORT).show();
        System.exit(0);
    }
    return;
}
```

2. 实现音乐简单的音乐列表，可以选择音乐列表中音乐进行播放
实现后界面如下图：



此时我们可以直接获取专辑图片，作为图片进行填充，之后我们可以获取音乐名字，此时就不一定需要时候文件名了。

- 第一步改造界面，没什么说的，加了一个 listview，因为只要显示歌曲名称，所以从简 ArrayAdapter 作为适配器和 Listview 自带的点击事件。在之前的点击函数中增加如下，pos 记录点击位置，然后 sta 改为 5 用于 Handler 更新 UI 和与 Service 通信更改歌曲时使用。

```
listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) {//listview 点击
        pos=i;sta=5;
    }
});
```

- 想要做出一个可以更改歌曲的界面需要从获取媒体库中歌曲信息，也就是内置 SD 卡中所有的歌曲信息，此处可以直接查询媒体库获取需要的信息。

```
/**
 * 获取媒体数据库
 */
public void scanAllAudioFiles(){
    //查询媒体数据库
    Cursor cursor = getContentResolver().query(MediaStore.Audio.Media.EXTERNAL_CONTENT_URI, null, null, null,
MediaStore.Audio.Media.DEFAULT_SORT_ORDER);
    //遍历媒体数据库
    if(cursor.moveToFirst()) {
        while (!cursor.isAfterLast()) {
            //歌曲编号
            int id = cursor.getInt(cursor.getColumnIndexOrThrow(MediaStore.Audio.Media._ID));
            //歌曲名
            String tilte = cursor.getString(cursor.getColumnIndexOrThrow(MediaStore.Audio.Media.TITLE));
            //歌曲专辑编号
            int album_id = cursor.getInt(cursor.getColumnIndexOrThrow(MediaStore.Audio.Media.ALBUM_ID));
            //歌曲文件的路径 ：MediaStore.Audio.Media.DATA
            String url = cursor.getString(cursor.getColumnIndexOrThrow(MediaStore.Audio.Media.DATA));
            //歌曲文件的大小 ：MediaStore.Audio.Media.SIZE
            Long size = cursor.getLong(cursor.getColumnIndexOrThrow(MediaStore.Audio.Media.SIZE));
            if (size > 1024 * 800) {//如果文件大小大于 800K，将该文件信息存入
                PATHS.add(url);
                Music_name.add(tilte);
                Music_id.add(String.valueOf(id));
                Music_albumid.add(String.valueOf(album_id));
            }
```

```
            cursor.moveToNext();
        }
    }
}
```

- 此处我们在 Service 和主界面第一次连接时初始化修改为

```
default: {//其他 code 设置 为初始化并回传数据
    Music_init();
    reply.writeInt(mediaPlayer.getDuration());
    reply.writeStringList(Music_name);
    reply.writeStringList(Music_id);
    reply.writeStringList(Music_albumid);
    break;
}
```

此处的 Music_init 为

```
/**
 * 歌曲库初始化，默认先使用第一首歌曲
 */
public void Music_init() {
    scanAllAudioFiles();
    try {
        mediaPlayer.setDataSource(PATHS.get(0));
        mediaPlayer.prepare();
        mediaPlayer.setLooping(true);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

- 之后我们在在 onServiceConnected 重载中增加如下，获取返回给主界面的歌曲名称和 ID 以及唱片 ID，用歌曲名称填充 lisview，在建立连接初始化之后加入读取返回的数据然后显示在主界面上，此处读取专辑图片用了网上现成的函数。

```
reply.readStringList(Music_name);
reply.readStringList(Music_id);
reply.readStringList(Music_albumid);
Bitmap bm = getArtwork(MainActivity.this,
Integer.valueOf(Music_id.get(0)),Integer.valueOf(Music_albumid.get(0)),true);
if(bm!=null)imageView1.setImageBitmap(bm);
listView.setAdapter(new ArrayAdapter<String>(MainActivity.this,
android.R.layout.simple_expandable_list_item_1,Music_name));
```

- 在 service 的 IBender 中增加一个状态：当点击某一项之后 mediaplayer 重新加载歌曲，然后重新初始化

```
**
 * 载入新的歌曲
 * @param pos
 */
private void Music_reInit(int pos){
    if (mediaPlayer != null) {
        mediaPlayer.reset();
        try {
            mediaPlayer.setDataSource(PATHS.get(pos));
            mediaPlayer.prepare();
            mediaPlayer.seekTo(0);
            mediaPlayer.setLooping(true);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

IBender 中加入新的状态

```
case 106: {
    Music_reInit(data.readInt());
    reply.writeInt(mediaPlayer.getDuration());
    break;
}
```

在 handler 中加入新的状态时的 UI 处理

```
else if (msg.what == 5) { //点击更换歌曲
    seekBar.setProgress(0);  //类似执行停止
    begin.setText(time.format(0));
    play.setText("PLAY");
    status.setText("");
    animator.setCurrentPlayTime(0); //动画初始化
    animator.cancel();
```

```
    animator1.setCurrentPlayTime(0);
    animator1.cancel();
    //获取图片
    Bitmap bm = getArtwork(MainActivity.this,
Integer.valueOf(Music_id.get(pos)),Integer.valueOf(Music_albumid.get(pos)),true);
    if(bm!=null)imageView1.setImageBitmap(bm);
    try {
        Parcel data = Parcel.obtain();
        Parcel reply = Parcel.obtain();
        data.writeInt(pos);
        myBinder.transact(106, data, reply, 0);
        int len = reply.readInt();
        seekBar.setMax(len); // 设置最大时长
        end.setText(time.format(len));//设置时间
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

六、 实验思考及感想

这次实验主要就是实现 Service 制作一个后台音乐播放器，主要的实现只有这里，实验主要也是为了理解如何使用 Service。使用中使用 Handler，还是在主线程中，只不过是调用了一个子线程让他定时更新 UI，虽然不用这个依然可以实现更新 UI 功能。

实验主要的困难还是在如何放入文件，因为虚拟机的版本问题，同时网上也没有什么正经的教程，都说是可以直接放进去的，直接放进去是可以放到内置 SD 卡上的不用别的步骤，直接放入，不过无法查看是否放入成功，因为那个目录打不开，而且想要放到指定文件夹比较麻烦，需要用 sdk 中的平台工具进行 root 才能做到上述两点，而网上教程基本都没有提及如何使用 root，这里的时间花销比较大。

至于 Service 的实现参考了一下实验文档给出的架构，然后把 MediaPlaying 所需的几个功能写好直接调用就行，连接的上课也讲得很清楚同时也不难。Handler 的实现也不难，其实就是把之前编写中点击事件更新 UI 的做法搬到 Handler。