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：

年级	15	专业（方向）	移动互联网
学号	15352155	姓名	赖贤城
电话	13727024851	Email	754578682@qq.com
开始日期	2017/12/3	完成日期	2017/12/3

一、 实验题目

服务与多线程--简单音乐播放器

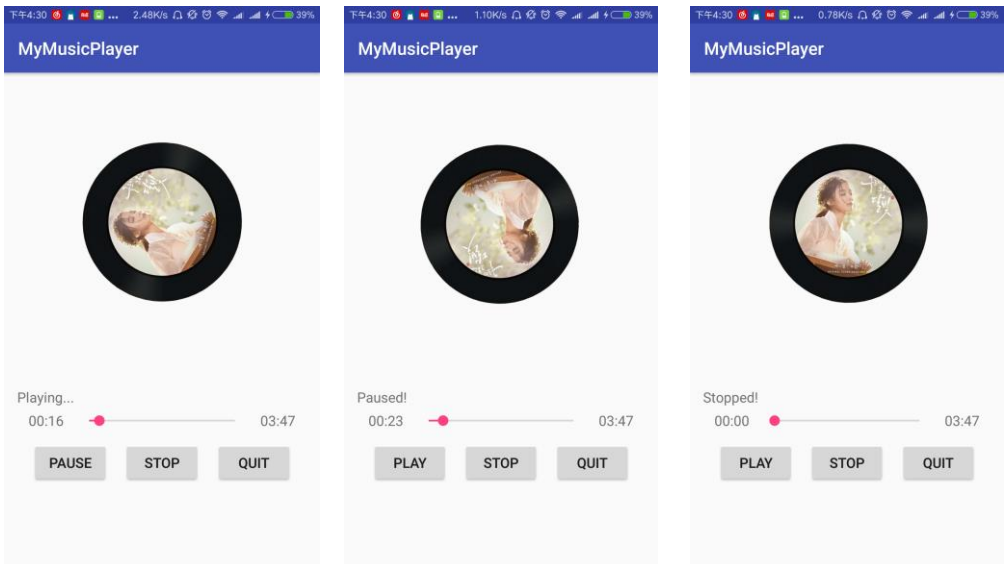
二、 实现内容

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

三、 课堂实验结果

（1） 实验截图



（2） 实验步骤以及关键代码

1. 定义播放界面布局

由于本实验重点不是布局，因此这里不展开。主要就是通过 linnearLayout 和 ConstraintLayout 进行布局

```
<receiver android:name=".widget.ShoppingWidget">
    <intent-filter>
        <action android:name="android.appwidget.action.APPWIDGET_UPDATE"/>
        <!-- action android:name="com.lxc.my.WIDGET_ADDITION"/ -->
        <action android:name="com.lxc.my.LAUNCH_WIDGET"/>
        <action android:name="com.lxc.action.widgetInitClick"/>
    </intent-filter>

    <meta-data
        android:name="android.appwidget.provider"
        android:resource="@xml/shopping_widget_info"/>
</receiver>
```

2. 定义 MusicService，用于播放音乐

重要的是 onBind 函数，该函数在 Service 被绑定的时候调用，然后返回一个 Binder 对象，Binder 是活动和服务通信的桥梁，这里返回自定义的一个 Binder（将在下面解释），另外这里还对 MediaPlayer 做了一些初始化操作，为下一步的播放做准备

```
public class MusicService extends Service {
    private MusicBinder musicBinder = new MusicBinder();
    private MediaPlayer mp = new MediaPlayer();
    public enum StateEnum {PLAY,PAUSE,STOP}
    private StateEnum curState;
    public MusicService() {
    }
    @Override
    public IBinder onBind(Intent intent) {
        try {
            mp.setDataSource("/storage/emulated/0/mymusic/任然 - 你好陌生人.mp3");
            mp.prepare();
            mp.setLooping(true);
        } catch (IOException e) {
            e.printStackTrace();
        }
        return musicBinder;
    }
```

上图的 MusicBinder 定义如下，它能处理 activity 发过来的消息，播放、暂停、停止、完全退出，这些操作直接通过调用 MediaPlayer 的相应函数处理。另外还有返回关于音乐的数据，这个通过将数据写入 reply 中传递过去，这样 activity 那边就可以从 reply 中将数据读出来

```
class MusicBinder extends Binder{
    @Override
    protected boolean onTransact(int code, Parcel data, Parcel reply, int flags) throws RemoteException {
        switch (code){
            //暂停或者播放
            case 1:
                if (mp.isPlaying()){
                    mp.pause();
                    curState = StateEnum.PAUSE;
                }
                else{
                    mp.start();
                    curState = StateEnum.PLAY;
                }
                break;
            //停止音乐
            case 2:
                mp.seekTo( msec: 0);
                mp.pause();
                curState = StateEnum.STOP;
                break;
            //完全退出
            case 3:
                mp.stop();
                mp.release();
                stopSelf();
                break;
            //拖动滑动条
            case 4:
                float progress = data.readFloat();
                mp.seekTo(((int)(progress*mp.getDuration())));
                break;
            //获取现在的音乐位置以及音乐总长度
            case 5:
                int curTime = mp.getCurrentPosition();
                int length = mp.getDuration();
                reply.writeInt(curTime);
                reply.writeInt(length);
                reply.setDataPosition(0);//设置偏移量为0，待会读的时候就能从头读
```

另外 Binder 里面记录了状态，定义了一个 get 函数返回状态，这个状态用于 activity 中根据状态初始化界面

```
public StateEnum getState(){
    return curState;
}
```

3. 播放，暂停，停止以及完全退出的逻辑

a. 点击事件处理

在点击播放，暂停，停止以及完全退出的时候调用 transact 函数与 service 通信，这里的参数 code 与 Binder 中的处理是一一对应的，上面已经解释了 Binder 会去播放音乐等操作。与 service 通信之后调用 updateViews 函数更新界面

```
public void onClick(View v) {
    switch (v.getId()){
        case R.id.bt_play:
            try {
                musicBinder.transact( code: 1, Parcel.obtain(), Parcel.obtain(),  flags: 0);
            } catch (RemoteException e) {
                e.printStackTrace();
            }
            if (isPause)
                updateViews(StateEnum.PLAY);
            else
                updateViews(StateEnum.PAUSE);
            break;
        case R.id.bt_stop:
            //只有正在播放的时候才stop
            if (musicBinder.getState() != MusicService.StateEnum.PLAY)
                return;
            try {
                musicBinder.transact( code: 2, Parcel.obtain(), Parcel.obtain(),  flags: 0);
                updateViews(StateEnum.STOP);
            } catch (RemoteException e) {
                e.printStackTrace();
            }
            break;
        case R.id.bt_quit:
            unbindService(serviceConnection);
            try {
                musicBinder.transact( code: 3, Parcel.obtain(), Parcel.obtain(),  flags: 0);
            } catch (RemoteException e) {
                e.printStackTrace();
            }
            finish();
            System.exit( status: 0);
            break;
    }
```

b. *点击后的界面更新与动画相关逻辑*

上面的 updateViews 函数定义如下，它主要是设置界面上的文字以及开启和关闭旋转动画，需要注意的是，这里暂停时的动画操作是关闭之后还给原 view 设置了一个旋转角度，这样就不会跳到其他的角度导致跳跃感，而如果是 stop 的话就将角度设置回 0 度，这样就回到了初始的状态

```
/**
 * 根据不同状态更新界面
 * @param state 新的状态
 */
private void updateViews(StateEnum state) {
    if (state == StateEnum.PLAY){
        isPause = false;
        btPlay.setText("Pause");
        tvHint.setText("Playing...");
        startRotationAnim();
    }
    else if (state == StateEnum.PAUSE){
        isPause = true;
        btPlay.setText("Play");
        tvHint.setText("Paused!");
        if (rotationAnimator != null && rotationAnimator.isRunning()){
            float cur_rotation = (Float) rotationAnimator.getAnimatedValue();
            rotationAnimator.end();
            imageView.setRotation(cur_rotation);//让角度停在停的时候
        }
    }
    else if (state == StateEnum.STOP){
        tvHint.setText("Stopped!");
        isPause = true;
        btPlay.setText("Play");
        if (rotationAnimator != null && rotationAnimator.isRunning())
            rotationAnimator.end();
            imageView.setRotation(0);//让角度恢复0度
    }
}
```

上面的代码中有一个函数是 startRotationAnim 函数，实现如下，主要就是使用 ObjectAnimator 去改变 view 的角度值，这样就很简单的实现了动画效果

```
/**
 * 开启旋转动画
 */
private void startRotationAnim() {
    //这里的360如果不加上imageView.getRotation()的话动画重复的时候会产生跳跃
    rotationAnimator = ObjectAnimator.ofFloat(imageView, propertyName: "rotation",
            ...values: imageView.getRotation(),360f+imageView.getRotation());
    rotationAnimator.setDuration(10000);
    rotationAnimator.setRepeatCount(ValueAnimator.INFINITE);
    rotationAnimator.setInterpolator(new LinearInterpolator());
    rotationAnimator.start();
}
```

4. 处理滑动条拖动的逻辑

实现 OnSeekBarChangeListener，重写 onStopTrackingTouch 函数，表示在用户停止拖顶的时候执行下述操作，这里是将当前的滑动条对应的比例传送给 service，让它去调整音乐的当前播放点，从而达到拖动滑动条就能调整音乐的效果

```
@Override
public void onStopTrackingTouch(SeekBar seekBar) {
    try {
        Parcel parcel = Parcel.obtain();
        parcel.writeFloat( val: ((float)seekBar.getProgress())/100);
        musicBinder.transact( code: 4,parcel , Parcel.obtain(), flags: 0);
    } catch (RemoteException e) {
        e.printStackTrace();
    }
}
```

5. 实时更新当前播放时间以及滑动条对应比例

a. 实现 Runnable 接口，作为线程执行块

在 run 函数中以一个 while 死循环作为最外层，每次循环都 sleep 一秒，这样线程就不会自己结束，然后内部逻辑是通过 Binder 从 service 那边得到音乐的数据，将数据包装之后发送给 Handler 从而转到 UI 线程去更新界面

```
/**
 * 每一秒和service通信一次，获取届时数据，然后交给hander更新界面
 */
@Override
public void run() {
    while(true){
        try {
            Parcel timeReply = Parcel.obtain();
            musicBinder.transact( code: 5, Parcel.obtain(), timeReply, flags: 0);
            int curTime = timeReply.readInt();
            int length = timeReply.readInt();
            Message message = new Message();
            message.what = 666;
            message.arg1 = ((int)(curTime*1.0/length*100));
            message.arg2 = curTime;
            progressHandler.sendMessage(message);

            Thread.sleep( millis: 1000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        } catch (RemoteException e) {
            e.printStackTrace();
        }
    }
}
```

b. Handler 更新界面

Handler 接到子线程传过来的数据之后就可以用这些数据对滑动条和当前时间的界面进行更新

```
/**
 * 线程获得数据之后，由Hander更新界面
 */
class  ProgressHandler extends Handler{
    @Override
    public void handleMessage(Message msg) {
        super.handleMessage(msg);
        if (msg.what == 666){
            sbMusic.setProgress(msg.arg1);
            int curTime = msg.arg2;
            tvCurTime.setText(formatter.format(new Date(curTime)));
        }
    }
}
```

6. 实现 ServiceConnection 接口

主要是写 onServiceConnected，其会在 activity 与 service 绑定的时候回调，其中的 Ibinder 对象就是 service 中返回的那个对象。当绑定成功的时候，我们就可以开启界面更新的线程以及监听滑动条了，因为这两个都是要和 service 通信的，所以要在绑定之后才能开启。

另外，注意这里还去获取了当前状态做对应的界面更新，这是因为有可能 app 关闭之后重新打开，但是音乐是能后台播放的，那么这时候音乐的状态有三种可能，所以要做三种界面上的处理

```
private ServiceConnection serviceConnection = new ServiceConnection() {
    @Override
    public void onServiceConnected(ComponentName name, IBinder service) {
        musicBinder = (MusicService.MusicBinder) service;

        //因为progressChangeListener里面要和service通信，所以绑定之后开始监听
        progressChangeListener progressListener = new progressChangeListener(musicBinder);
        sbMusic.setOnSeekBarChangeListener(progressListener);

        //因为线程里面要和service通信，所以绑定之后才开启线程
        updateThread = new Thread(new connectRunnable(musicBinder, progressHandler));
        updateThread.start();

        //获得音乐总时间
        Parcel timeReply = Parcel.obtain();
        try {
            musicBinder.transact( code: 5, Parcel.obtain(), timeReply,  flags: 0);
        } catch (RemoteException e) {
            e.printStackTrace();
        }
        timeReply.setDataPosition(4);//第二个数（length）的位置，因为int是4字节
        int musicLength = timeReply.readInt();
        tvWholeTime.setText(formatter.format(new Date(musicLength)));

        //获取当前状态，根据当前状态更新界面
        //原因：有可能app关闭之后重新打开，因为音乐是能后台播放的，那么这时候音乐的状态有这三种可能
        if (musicBinder.getState()==MusicService.StateEnum.PLAY){
            updateViews(StateEnum.PLAY);
        }
        else if (musicBinder.getState()==MusicService.StateEnum.PAUSE){
            updateViews(StateEnum.PAUSE);
        }
        else{
            updateViews(StateEnum.STOP);
        }
    }
```

7. 申请读取存储器的动态权限

首先，进入 app 的时候应该检查是否已经有权限（下图第 1 个函数）。 若是没有权限，那么请求权限，返回结果（下图第 2 个函数），如果用户同意了，那么就可以开启服务播放音乐（下图第 3 个函数），否则直接关闭

```
public void requestPermission() {
    if (ContextCompat.checkSelfPermission(mainActivity,
            Manifest.permission.READ_EXTERNAL_STORAGE) != PackageManager.PERMISSION_GRANTED){
        ActivityCompat.requestPermissions(mainActivity, new String[]{Manifest.permission.READ_EXTERNAL_
    }
    else{
        startAndBingService();
    }
}


public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] g
    if (grantResults.length>0 && grantResults[0]==PackageManager.PERMISSION_GRANTED){
        startAndBingService();
    }
    else{
        Toast.makeText(mainActivity, text: "拒绝权限，无法使用程序",Toast.LENGTH_SHORT).show();
        mainActivity.finish();
    }
}

private void startAndBingService() {
    Intent serviceIntent =  new Intent(mainActivity, MusicService.class);
    mainActivity.startService(serviceIntent);
    mainActivity.bindService(serviceIntent, mainActivity.getServiceConnection(), BIND_AUTO_CREATE);
}
```

（3） 实验遇到困难以及解决思路

1. pause 的时候动画结束后回到初始状态

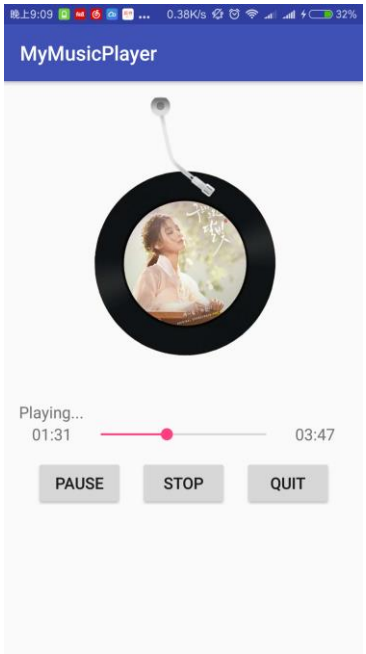
解决：本来想着是不是有什么 api 设置动画结束的时候就停在当前的状态，然后就想开始查 api，后来一想我傻啊，直接设置 view 的旋转角就好了啊，然后就解决了

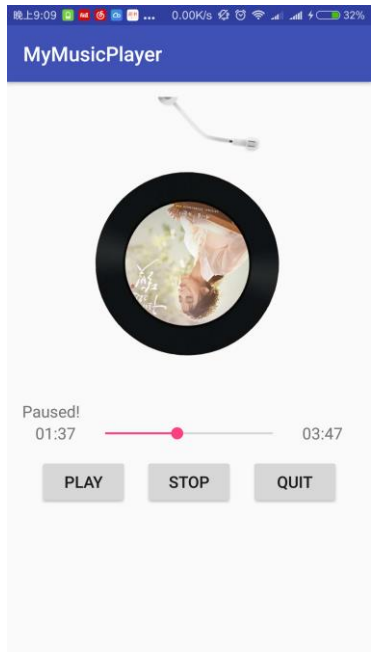
```
else if (state == StateEnum.PAUSE){
    isPause = true;
    btPlay.setText("Play");
    tvHint.setText("Paused!");
    if (rotationAnimator != null && rotationAnimator.isRunning()){
        float cur_rotation = (Float) rotationAnimator.getAnimatedValue();
        rotationAnimator.end();
        imageView.setRotation(cur_rotation);//让角度停在停的时候
    }
}
```

四、 课后实验结果

添加唱针以及对应的动画（仿网易云音乐）

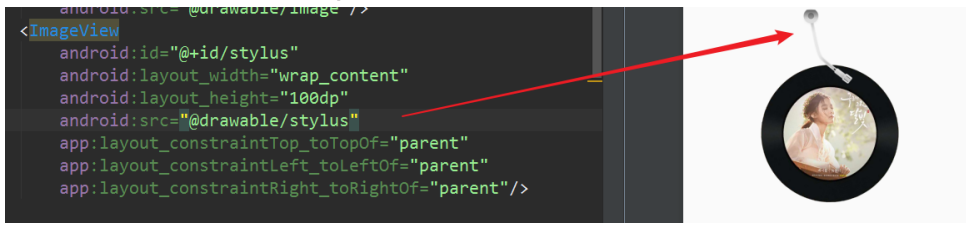
1. 结果展示





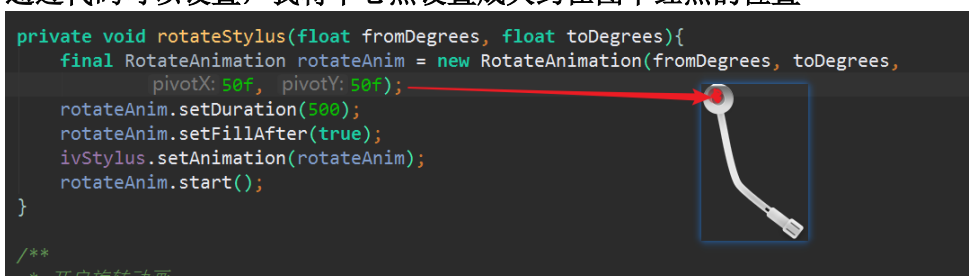
2. 实验过程

1. 布局中加入唱针对应的 imageview



2. 编写唱针旋转动画

这里的旋转和唱片的旋转不太一样，这里的旋转中心应该是左上角的地方，通过代码可以设置，我将中心点设置成大约在图中红点的位置



3. 设置暂停或停止的时候旋转出去，播放的时候旋转回来

```
private void updateViews(StateEnum state) {
    if (state == StateEnum.PLAY){
        isPause = false;
        btPlay.setText("Pause");
        tvHint.setText("Playing...");
        startRotationAnim();
        rotateStylus(-30f, 0f);
    }
    else if (state == StateEnum.PAUSE){
        isPause = true;
        btPlay.setText("Play");
        tvHint.setText("Paused!");
        if (rotationAnimator != null && rotationAnimator.isRunning()){
            float cur_rotation = (Float) rotationAnimator.getAnimatedValue();
            rotationAnimator.end();
            imageView.setRotation(cur_rotation);//让角度停在停的时候
            rotateStylus(ivStylus.getRotation(), -30f);
        }
    }
    else if (state == StateEnum.STOP){
        tvHint.setText("Stopped!");
        isPause = true;
        btPlay.setText("Play");
        if (rotationAnimator != null && rotationAnimator.isRunning())
            rotationAnimator.end();
            imageView.setRotation(0);//让角度恢复0度
            rotateStylus(ivStylus.getRotation(), -30f);
    }
```

五、 实验思考及感想

1. 主要是以前没怎么写过服务的相关代码，这次算是更熟悉了服务的相关概念和代码
2. 要多看文档，多学习 api，安卓已经提供了足够多的 api，懂得怎么将它们组合起来就能做出很不错的效果

作业要求：

1. 命名要求: 学号_姓名_实验编号，例如 15330000_林 XX_lab1。
2. 实验报告提交格式为 pdf。
3. 实验内容不允许抄袭，我们要进行代码相似度对比。如发现抄袭，按 0 分处理。