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 级	专业（方向）	软件工程（移动信息工程）
学号	15352223	姓名	刘朝开
电话	15626489095	Email	1209354383@qq.com
开始日期	2017.9.30	完成日期	2017.10.1

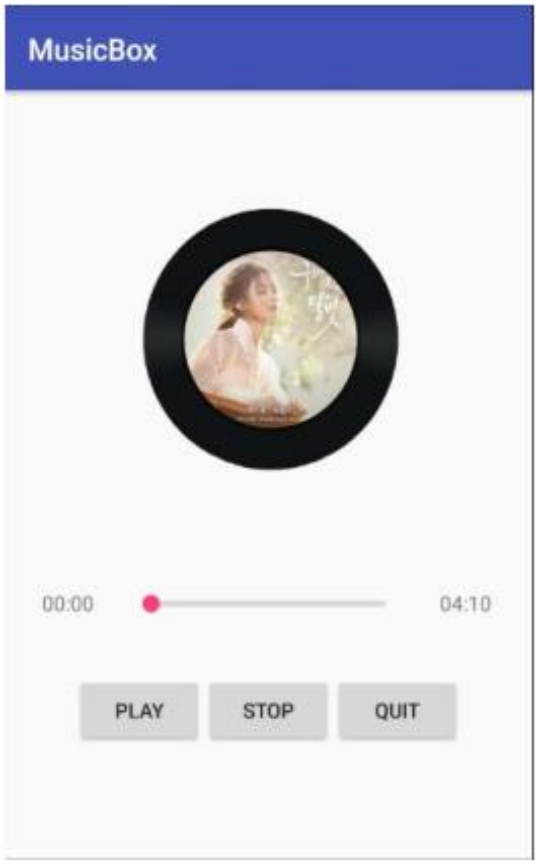
一、 实验题目

服务与多线程--简单音乐播放器

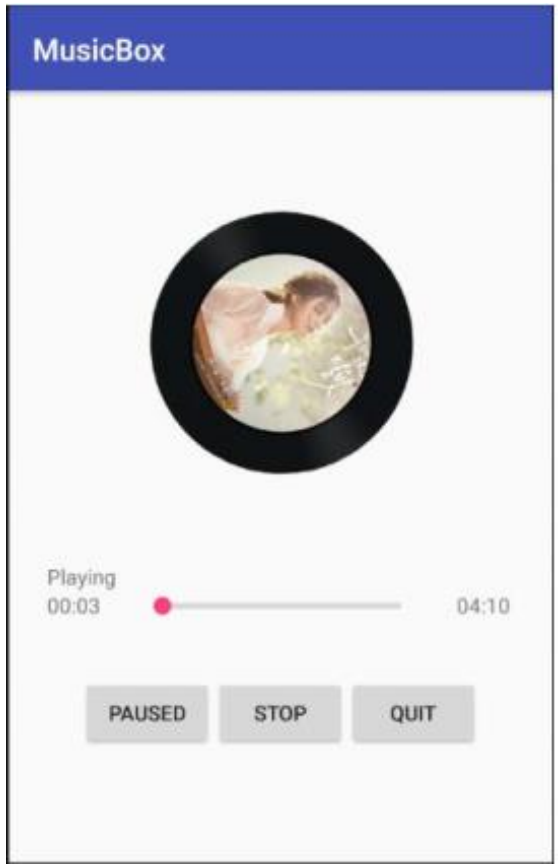
二、 实现内容

实现一个简单的播放器，要求功能有：

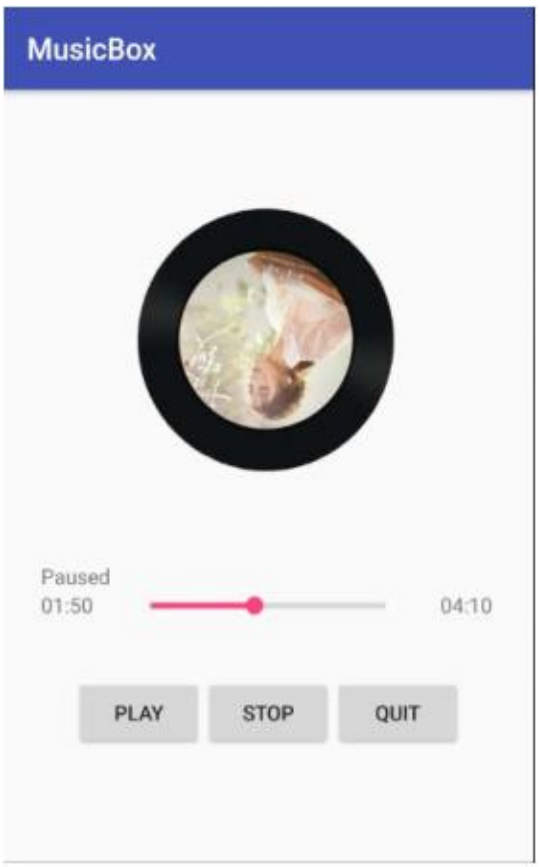
1. 播放、暂停，停止，退出功能；
2. 后台播放功能；
3. 进度条显示播放进度、拖动进度条改变进度功能；
4. 播放时图片旋转，显示当前播放时间功能；



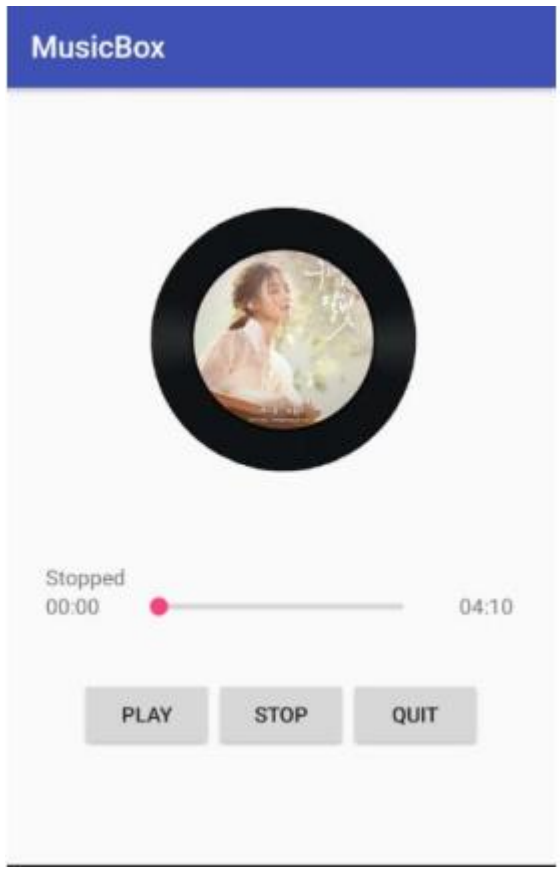
打开程序主页面



开始播放



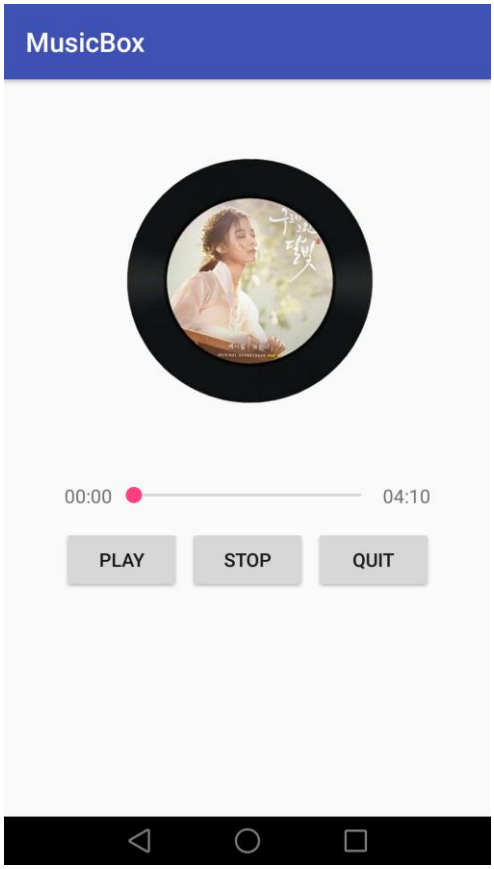
暂停



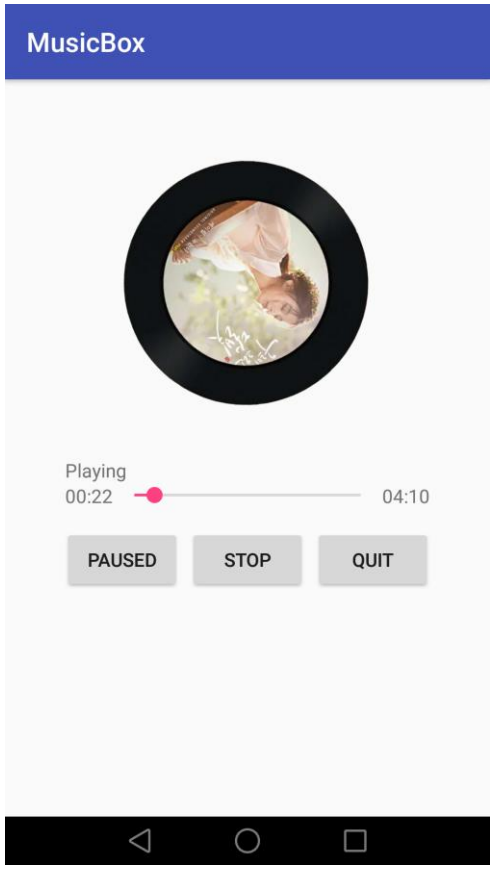
停止

三、 课堂实验结果

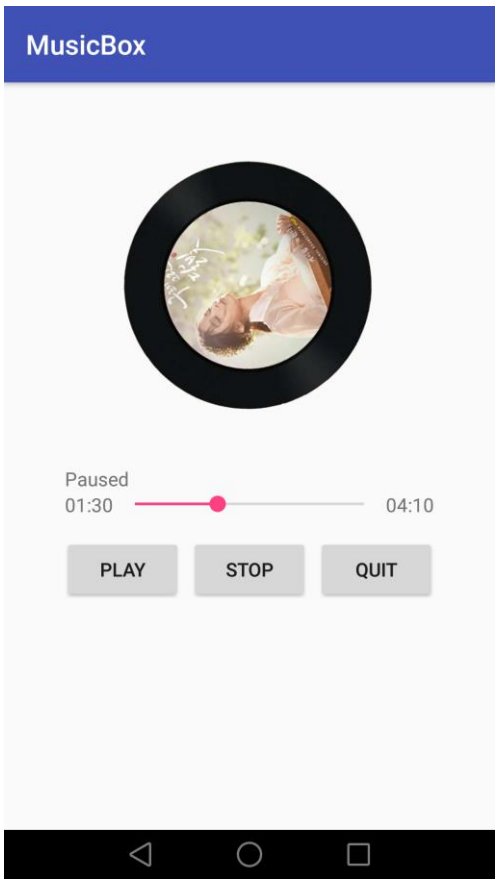
（1） 实验截图

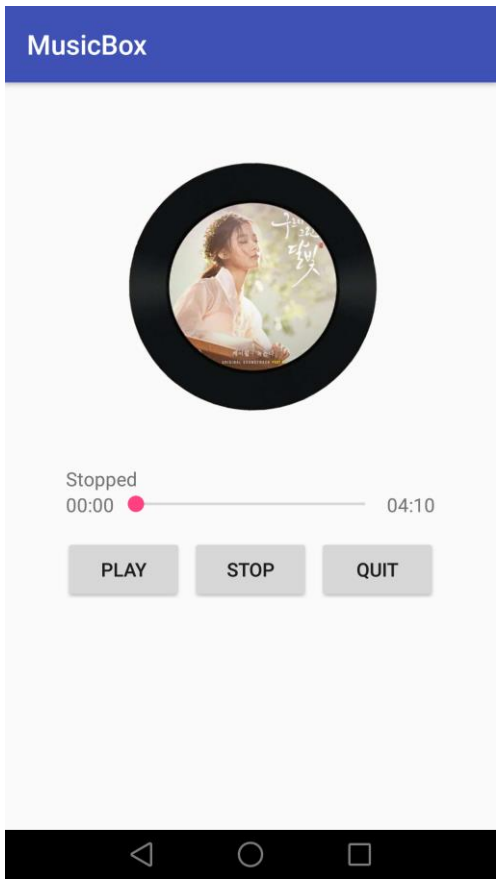


打开程序主页面



开始播放





暂停　　停止

（2） 实验步骤以及关键代码

1）设置布局文件；（代码较为简单，不贴图）

2）根据按钮使用写与文本变化相关的代码；（代码较为简单，与 3）一起）

3）设置图片动画效果；

首先定义了图片旋转的属性：0-359°旋转（360°会卡顿），线性匀速，旋转一圈时长 60s，不断重复（-1）。

```
//图片旋转动画
animator = ObjectAnimator.ofFloat(imageView, propertyName: "rotation", ...values: 0f, 359f);
LinearInterpolator interpolator = new LinearInterpolator();//线性
animator.setInterpolator(interpolator);//设置匀速旋转，在xml文件中设置会出现卡顿
animator.setDuration(60000);
animator.setRepeatCount(-1);
```

根据不同的按键设置文本，和动画效果。

```
private void initUI() {
    switch (state) {
        case "start":
            textViews[0].setText(null);
            buttons[0].setText("PLAY");
            break;
        case "play":
            buttons[0].setText("PAUSED");
            textViews[0].setText("Playing");
            if(animator.isPaused()) {
                animator.resume();
            }
            else if(!animator.isStarted()) animator.start();
            break;
        case "pause":
            buttons[0].setText("PLAY");
            textViews[0].setText("Paused");
            animator.pause();
            break;
        case "stop":
            textViews[0].setText("Stopped");
            buttons[0].setText("PLAY");
            animator.end();
            break;
        default:break;
    }
```

4）新建服务类，读取获得音乐文件，编写与按钮实现的开始，暂停和停止相关的代码。

读取音乐文件：

```
try{
    String filePath = Environment.getExternalStorageDirectory().getAbsolutePath()+"/Music/melt.mp3";
    mp.setDataSource(filePath);
    mp.prepare();
    mp.seekTo( msec: 0);
    mp.setLooping(true);
}
catch (Exception e) {
    e.printStackTrace();
}
return START_STICKY;
```

音乐的开始，暂停：

```
public  void playMusic(){
    if(mp != null) {
        if(mp.isPlaying()) {
            mp.pause();
            curState = "pause";
        }
        else {
            try {
                mp.start();
                curState = "play";
            } catch (Exception e){
                e.printStackTrace();
            }
        }
    }
}
```

音乐的停止：

```
public void stopMusic() {
    if(mp != null){
        mp.stop();
        curState = "stop";
        try {
            mp.prepare();
            mp.seekTo( msec: 0);
        }catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

5）定义内部类 MusicBinder 继承于 Binder，重写 onTransact 函数。用于 activity 与 service 之间的通信。

```
public class  MusicBinder extends Binder {
    @Override
    protected boolean onTransact(int code, Parcel data, Parcel reply, int flags) throws RemoteException {
        switch (code) {
            case 101://播放按钮，服务处理函数
                playMusic();
                reply.writeString(curState);
                break;
            case 102://停止按钮服务处理函数
                stopMusic();
                reply.writeString(curState);
                break;
            case 103://退出按钮，服务处理函数
                stopMusic();
                reply.writeString(curState);
                break;
            case 104://界面刷新，服务返回数据函数
                int mpCposition = mp.getCurrentPosition();
                int mpDuration = mp.getDuration();
                reply.writeInt(mpCposition);
                reply.writeInt(mpDuration);
                break;
            case 105://拖动进度条，服务处理函数
                mp.seekTo(data.readInt());
                break;
            default:break;
        }
        return super.onTransact(code,data,reply,flags);
    }
    public MusicService getService() {
        return MusicService.this;
    }
```

6）动态获取权限。

确认是否获得权限，若未获得就进行申请。否则标记 isrwsdcard 为 true。

```
private void verifyStoragePermission(Activity activity)
{
    try {
        int permission = ActivityCompat.checkSelfPermission(activity,
                permission: "android.permission.READ_EXTERNAL_STORAGE"
        );
        if(permission != PackageManager.PERMISSION_GRANTED) {
            ActivityCompat.requestPermissions(activity, new String[]
                    {Manifest.permission.READ_EXTERNAL_STORAGE}, requestCode: 1);
        }
        else {
            isrwsdcard = true;//记录可读可写内存卡
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

在申请结果处，如若申请成功则将 isrwsdcard 置为 true。

```
@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions,
                                       @NonNull int[] grantResults) {
    if(grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED){
        isrwsdcard = true;
    }
    else {
        System.exit( status: 0);
    }
}
```

在 onCreate 中，当申请权限成功之后，便可以启动和绑定服务。

```
initView();
verifyStoragePermission( activity: this);//动态申请权限
if(isrwsdcard)
{
    Musicintent = new Intent( packageContext: this,MusicService.class);
    startService(Musicintent);//开始服务
    bindService(Musicintent,connection,BIND_AUTO_CREATE);//绑定服务
}
```

7）在 Manactivity 中通过调用 transact 函数发送信息执行相应的按钮操作。

按钮点击事件中，传递对应的信息给服务的 onTransact 函数。

```
@Override
public void onClick(View view) {
    switch (view.getId())
    {
        case R.id.play:
            code = 101;
            data = Parcel.obtain();
            reply = Parcel.obtain();
            sendMessage(code,data,reply, flag: 0);
            reply.setDataPosition(0);
            state = reply.readString();
            break;
        case R.id.stop:
            code = 102;
            data = Parcel.obtain();
            reply = Parcel.obtain();
            sendMessage(code,data,reply, flag: 0);
            reply.setDataPosition(0);
            state = reply.readString();
            break;
```

```
            case R.id.quit:
                code = 103;
                data = Parcel.obtain();
                reply = Parcel.obtain();
                sendMessage(code,data,reply, flag: 0);
                unbindService(connection);
                stopService(Musicintent);
                try{
                    finish();
                    System.exit( status: 0);
                }catch (Exception e) {
                    e.printStackTrace();
                }
                break;
            default:break;
        }
    }
```

其中 sendMessage 为自定义的函数，实现如下：调用了 transact 函数传递数据。

```
private void sendMessage(int code, Parcel data, Parcel reply, int flag)//发送信息
{
    try {
        musicBinder.transact(code,data,reply, i1: 0);
    }catch (RemoteException e) {
        e.printStackTrace();
    }
}
```

通过步骤 5）的 onTransact 函数接收和处理信息，或返回信息。

8）定义子线程用于实时更新 UI。

子线程无法直接更新 UI,需要发送信息给主线程。

```
thread = new Thread()
{
    @Override
    public void run() {
        while(true)
        {
            try {
                Thread.sleep( millis: 100);
            }catch (InterruptedException e) {
                e.printStackTrace();
            }
            if(connection!=null)
            {
                Message message = new Message();
                message.what = 1;
                handler.sendMessage(message);
            }
        }
    }
};
thread.start();//开启线程
```

在 onCreate 函数中，通过 handler 接收子线程发送的信息，不断的刷新进度条和当前的播放时间。

```
handler = (Handler) handleMessage(msg) → {
        super.handleMessage(msg);
        switch (msg.what) {
            case 1:
                code = 104;
                data = Parcel.obtain();
                reply = Parcel.obtain();
                try {
                    musicBinder.transact(code,data,reply, i1: 0);
                }catch (RemoteException e) {
                    e.printStackTrace();
                }
                reply.setDataPosition(0);
                int getCposition = reply.readInt();
                reply.setDataPosition(4);
                int getDuratin = reply.readInt();
                if(isrwsdcard) {
                    textViews[1].setText(time.format(getCposition));
                    textViews[2].setText(time.format(getDuratin));
                    seekBar.setProgress(getCposition);
                    seekBar.setMax(getDuratin);
                }
                initUI();
                break;
            default:break;
```

9）进度条被拖动时，发送信息给 service,用于更新歌曲播放进度。

通过重写 seekBar 的 setOnSeekBarChangeListener 实现。

```
seekBar.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
    @Override
    public void onProgressChanged(SeekBar seekBar, int i, boolean b) {...}
    @Override
    public void onStartTrackingTouch(SeekBar seekBar) {...}

    @Override
    public void onStopTrackingTouch(SeekBar seekBar) {
        code = 105;
        data = Parcel.obtain();
        data.writeInt(seekBar.getProgress());
        reply = Parcel.obtain();
        try {
            musicBinder.transact(code,data,reply, i1: 0);
        }catch (RemoteException e) {
            e.printStackTrace();
        }
    }
});
```

（3） 实验遇到困难以及解决思路

困难：

1 由于使用的手机版本为 6.0 以上，没有动态申请权限，代码无法正常运行；

2.动画无法正常旋转。

3 按 back 键退出应用再次打开应用发现进度条，文本等被再次初始化。

解决思路：

1 动态申请权限，使用标记，如成功申请权限则将标记记为 true，开启和绑定服务。

2.使用 pause 和 resume 来对动画进行暂停和继续。判断动画！isStarted()和 isPaused()来区分调用动画的 start()函数还是 resume()函数。

3.重写 back 键的代码。如下：

```
@Override//重写back键，实现Home效果
public void onBackPressed() {
    Intent intent = new Intent(Intent.ACTION_MAIN);
    intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
    intent.addCategory(Intent.CATEGORY_HOME);
    startActivity(intent);
}
```

四、 课后实验结果

五 实验思考及感想

这次试验总体给我个人的感觉还是偏难，有些代码还是无法理解，对于线程部分的工作原理还是处于一种比较混乱的状态，关于服务与 Activity 之间的绑定和信息传递也显得比较模糊。虽然很多的内容在课上也有提到，但实际运用起来还是比较困难。往后还需对服务方面的知识尽量的去理解。

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。