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**

姓名	学号	班级	电话	邮箱
张子豪	15352427	15M1	15989046143	ahzzh1998@163.com

1.实验题目

服务与多线程--简单音乐播放器

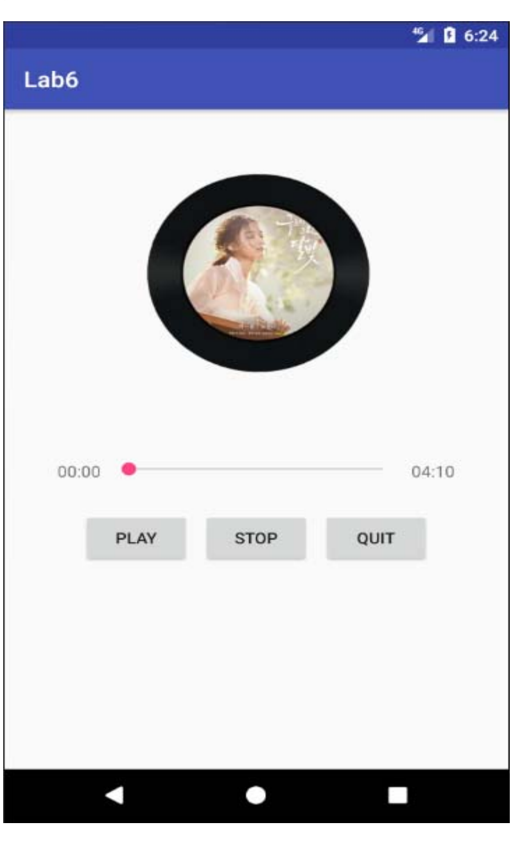
2.实现内容

实现一个简单的播放器，要求功能有：

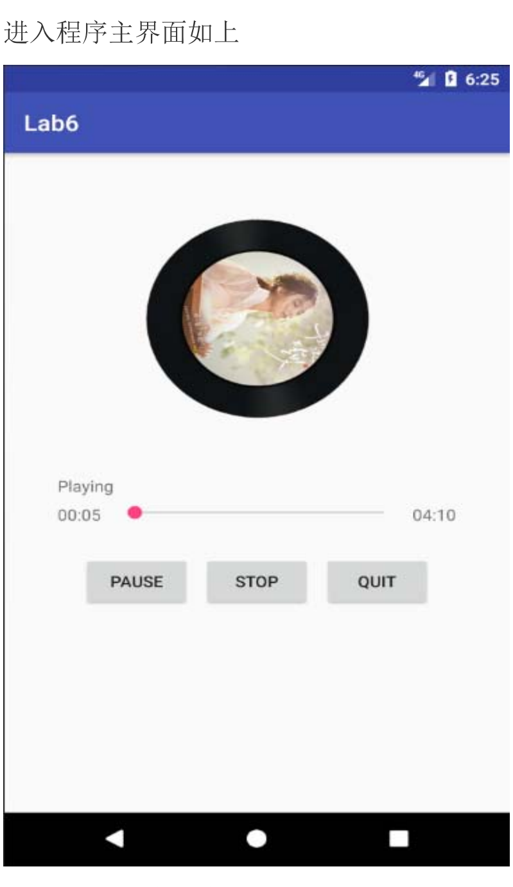
1. 播放、暂停，停止，退出功能；
2. 后台播放功能；
3. 进度条显示播放进度、拖动进度条改变进度功能；
4. 播放时图片旋转，显示当前播放时间功能 。

3.课堂实验结果

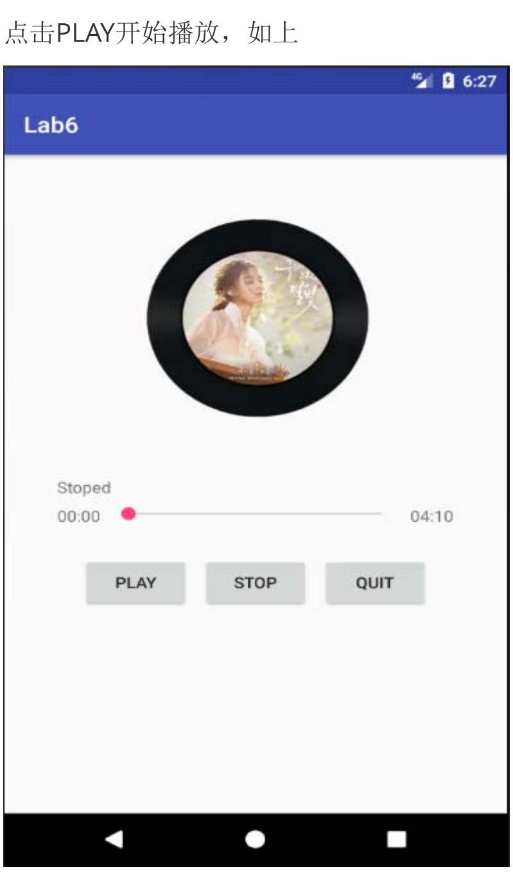
- 实验截图



进入程序主界面如上



点击PLAY开始播放，如上



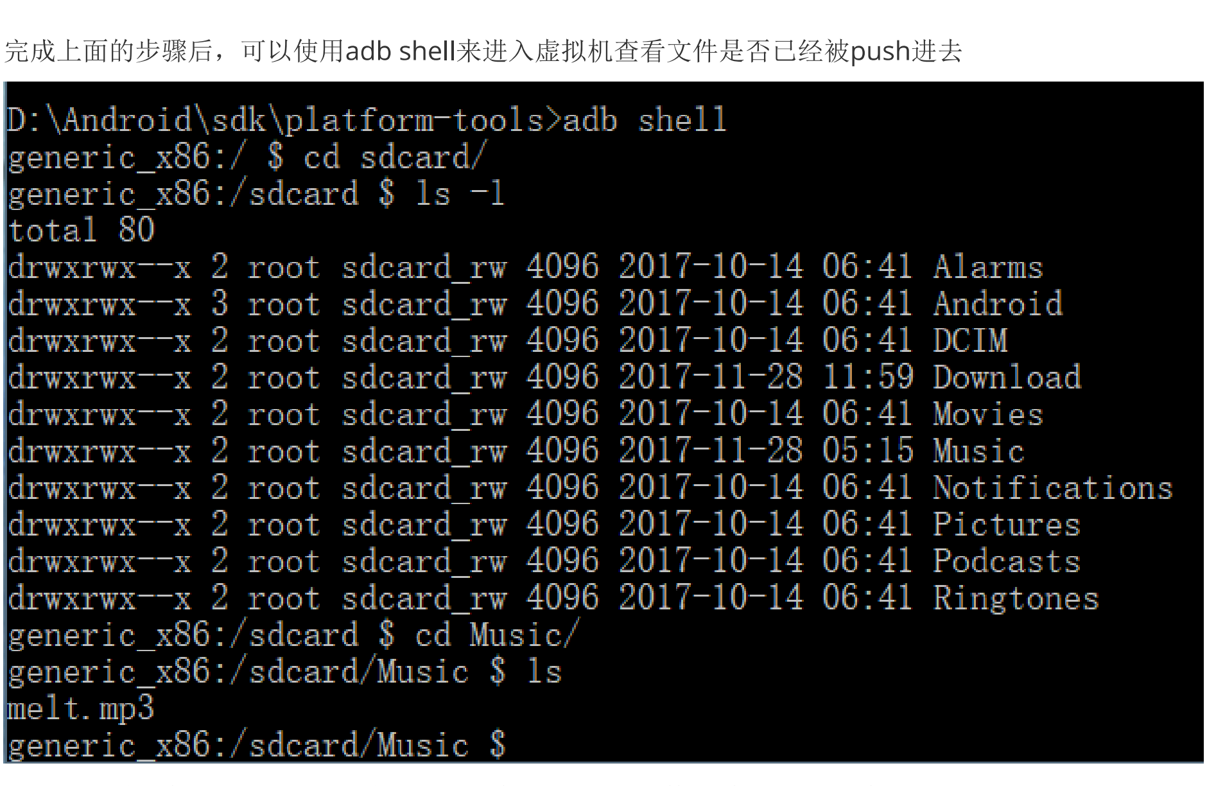
点击STOP，音乐停止，进度条归0，图片回正，如上

- 实验步骤以及关键代码
  - 将音乐文件导进虚拟机

    要完成这次实验首先需要把文件放进虚拟机，高系统的虚拟机可能需要权限，使用Android Studio的adb程序可以完成这个步骤。data文件夹需要的权限比较复杂，因此我选择将文件放进内置sd卡。

```
命令行进入D:\sdk\platform-tools目录
adb root    //root虚拟机，获得push文件的权限
adb push melt.mp3 /sdcard/Music   //将文件放进虚拟机/sdcard/Music目录下
```

    完成上面的步骤后，可以使用adb shell来进入虚拟机查看文件是否已经被push进去



```
D:\Android\sdk\platform-tools>adb shell
generic_x86:/ $ cd sdcard/
generic_x86:/sdcard $ ls -l
total 80
drwxrwx--x 2 root sdcard_rw 4096 2017-10-14 06:41 Alarms
drwxrwx--x 3 root sdcard_rw 4096 2017-10-14 06:41 Android
drwxrwx--x 2 root sdcard_rw 4096 2017-10-14 06:41 DCIM
drwxrwx--x 2 root sdcard_rw 4096 2017-11-28 11:59 Download
drwxrwx--x 2 root sdcard_rw 4096 2017-10-14 06:41 Movies
drwxrwx--x 2 root sdcard_rw 4096 2017-11-28 05:15 Music
drwxrwx--x 2 root sdcard_rw 4096 2017-10-14 06:41 Notifications
drwxrwx--x 2 root sdcard_rw 4096 2017-10-14 06:41 Pictures
drwxrwx--x 2 root sdcard_rw 4096 2017-10-14 06:41 Podcasts
drwxrwx--x 2 root sdcard_rw 4096 2017-10-14 06:41 Ringtones
generic_x86:/sdcard $ cd Music/
generic_x86:/sdcard/Music $ ls
melt.mp3
generic_x86:/sdcard/Music $
```

    这里可以看到在/sdcard/Music路径下已经有melt.mp3文件，说明push成功。

  - 创建Service并与Activity通信

    这次实验我们需要使用Servive来实现音乐播放并与Activity进行通信，因此需要重写onCreate方法与onBind方法。onCreate代码如下

```
public void onCreate(){ //service第一次创建时执行
    super.onCreate();
    Log.i("service","service已创建");
    try{
        File directory_music = new File(Environment.getExternalStorageDirectory(),
"Music");  //获得路径/sdcard/Music
        mediaPlayer.setDataSource(directory_music+"/melt.mp3");//获得文件路径
        mediaPlayer.prepare();
        mediaPlayer.setLooping(true);
        MainActivity.seekBar.setMax(mediaPlayer.getDuration());//设置seekbar的最大值
        MainActivity.min=0;
        MainActivity.second=mediaPlayer.getDuration();//读取音乐文件时长并计算其分、秒
        MainActivity.second/=1000;
        while (MainActivity.second>=60){
            MainActivity.second-=60;
            MainActivity.min++;
        }
    } catch (Exception e){
        e.printStackTrace();
    }
    //根据上面计算出的分、秒来更新界面中的音乐总时长的TextView
    if(MainActivity.second<10)
MainActivity.end_time.setText("0"+MainActivity.min+":0"+MainActivity.second);
    else MainActivity.end_time.setText("0"+MainActivity.min+":"+MainActivity.second);
}
```

    onBind方法如下，返回Service类中的一个成员myBinder

```
public IBinder onBind(Intent intent) { return myBinder; }
```

    考虑到要使用Binder让Service与Activity进行通信，因此需要自定义一个MyBinder类并重写onTransact方法，根据传入参数中的code的不同值来执行不同的操作。

```
public class MyBinder extends Binder {
    @Override
    protected boolean onTransact(int code, Parcel data, Parcel reply, int flags) throws
RemoteException {
        if(code==101){ //播放/暂停按钮
            if(mediaPlayer.isPlaying()){
                mediaPlayer.pause();
                MainActivity.isPlaying=false;
            }
            else{
                mediaPlayer.start();
                MainActivity.isPlaying=true;
            }
        }
        else if(code==102){ //停止按钮
            Log.i("停止","102");
            MainActivity.seekBar.setProgress(0);
            MainActivity.begin_time.setText("00:00");
            MainActivity.isPlaying=false;
            if(mediaPlayer!=null){
                mediaPlayer.stop();
                try{
                    mediaPlayer.prepare();
                } catch (Exception e){
                    e.printStackTrace();
                }
            }
        }
        else if(code==103){ //退出按钮
            System.exit(0);
        }
        else if(code==104){ //停止拖动进度条
            mediaPlayer.seekTo(MainActivity.seekBar.getProgress());
        }
        return super.onTransact(code,data,reply,flags);
    }
}
```

    开启Service并绑定

```
sc=new ServiceConnection() {
    @Override
    public void onServiceConnected(ComponentName name, IBinder service) {
        myBinder=(MyService.MyBinder) service;
    }

    @Override
    public void onServiceDisconnected(ComponentName name) {
        sc=null;
    }
};
Intent service_intent=new Intent(this,MyService.class);
startService(service_intent);
bindService(service_intent,sc, BIND_AUTO_CREATE);
```

  - 进度条的实现

    首先在播放时需要每秒同步进度条，创建一个子线程，在死循环中执行以下步骤：

    > 1. 休眠一秒
    > 2. 更新seekbar

```
Thread thread=new Thread(){
    @Override
    public void run(){
        while (true){
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e){
                e.printStackTrace();
            }
            if(isPlaying&&!isProcessChange){ //音乐正在播放且进度条没有被拖动，则更新进度条
                mHandler.obtainMessage(101).sendToTarget();//使用mHandler传递信息
            }
        }
    }
};
```

    mHandler中更新进度条

```
final Handler mHandler=new Handler(){
    @Override
    public void handleMessage(Message msg){
        super.handleMessage(msg);
        if(msg.what==101){
            seekBar.setMax(MyService.mediaPlayer.getDuration());
            int now_time=MyService.mediaPlayer.getCurrentPosition();
            MainActivity.seekBar.setProgress(now_time);//更新进度条
            int now_second=now_time/1000,now_min=0;//更新TextView
            while (now_second>=60){
                now_second-=60;
                now_min++;
            }
            if(now_second<10)
MainActivity.begin_time.setText("0"+now_min+":0"+now_second);
            else MainActivity.begin_time.setText("0"+now_min+":"+now_second);
        }
    }
};
```

    拖动进度条改变进度，监听seekbar是否改变

```
seekBar.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
    @Override
    public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {
        if(fromUser){ //进度由人为改变
            isProcessChange=true; //使能置true，不再每秒根据音乐进度更新进度条
            int now_time=progress;
            int now_second=now_time/1000,now_min=0;
            while (now_second>=60){
                now_second-=60;
                now_min++;
            }
            if(now_second<10)
MainActivity.begin_time.setText("0"+now_min+":0"+now_second);
            else MainActivity.begin_time.setText("0"+now_min+":"+now_second);
        }
    }

    @Override
    public void onStartTrackingTouch(SeekBar seekBar) {

    }

    @Override
    public void onStopTrackingTouch(SeekBar seekBar) { //进度条拖动停止
        isProcessChange=false;//使能置false，每秒根据音乐进度更新进度条
        try{
            //与Service通信，根据当前进度条更新MediaPlayer
            myBinder.onTransact(104,Parcel.obtain(),Parcel.obtain(),0);
        } catch (RemoteException e){
            e.printStackTrace();
        }
    }
});
```

  - 图片旋转动画

    > 1. 播放时开始旋转
    > 2. 暂停时停止旋转，继续播放时从停止位置继续旋转
    > 3. 停止时图片回正

    动画设计需要满足以上三个要求。使用ObjectAnimator类，暂停时记录下当前的角度，下次旋转从记录下的角度开始旋转即可完成第二点；让图片从0开始旋转然后立刻cancel，就可以完成图片回正。具体代码如下

```
//播放时开始旋转
animator=ObjectAnimator.ofFloat(imageView,"rotation",degree,360+degree);
animator.setDuration(5000);
animator.setRepeatCount(-1); //动画重复进行
animator.setInterpolator(new LinearInterpolator()); //动画播放时间线性分布
animator.start();
```

```
//暂停时停止旋转
degree=(Float) animator.getAnimatedValue(); //获取当前旋转到的角度
animator.cancel();
```

```
//停止时图片回正
degree=0;
animator.cancel();
animator=ObjectAnimator.ofFloat(imageView,"rotation",degree,360+degree);
animator.start();
animator.cancel();
```

  - 动态申请读取权限(雾)

    由于我没有安卓手机，虚拟机版本也比较低不需要申请权限，所以这里只有个人思路。。不过是已经在其他同学的手机上实践过可行的（其实也是因为有同学来问我这一步怎么做才想了一下思路，所以顺便写进了报告里 -,-）。

    申请权限的代码博客上一大堆，实验文档里也有给出，这里不再赘述，动态读取权限可能会出现的一个问题就是，使用**用户给予权限后再进行开启Service等活动**的做法的话，第一次进入app并给予权限能够正常运行，但是之后再进入app时，由于已经有了权限，因此不会再进入到onRequestPermissionResult函数中，也就无法开启Service。

    一个解决方案是在MainActivity的onCreate过程中检测是否有权限，如果有的话则给bool类型的hasPermission变量赋值true，否则弹出请求框请求用户给予权限。然后创建一个循环运行的子线程，检测当前是否有权限进行读写，如果有的话则开启Service进行播放器的初始化，完成后结束这个子线程即可。

```
Thread thread=new Thread(){
    @Override
    public void run(){
        while (true){
            if (hasPermission){ //拥有权限
                //开启Service
                //...
                //...
                break;
             }
        }
    }
};
```

- 实验遇到困难以及解决思路
  1. 按照实验文档中的做法读取不到音乐文件。

     解决思路：由于不同的虚拟机系统版本不一样，加之data文件夹的读写权限比较复杂，很麻烦，因此将文件放进内置sd卡比较方便。这样也可以用`Environment.getExternalStorageDirectory()`函数直接读取到路径。

  2. 验收的时候要求**按下物理返回键后再进入app动画仍在继续运行** 。

     由于这个实验做两周，因此第一周去验收的时候才知道有这个要求。

     解决思路：个人认为有两种解决方法，一种是将动画相关的代码丢进Service里，这样就跟音乐播放一样在后台也能运行，不过这样的话在我原有的代码基础上改动较大，因此我选择了另一种方法：

```
@Override
public void onBackPressed(){
    moveTaskToBack(false);
}
```

     直接选择按下物理返回键后后台运行整个程序，这样就不会出问题了。

4.实验思考及感想

经过这次实验，我对Service与多线程有了一定了解。Service可以在后台运行，这样将需要在后台运行的功能放进Service里，而不用将整个app放进后台，能够节省大量内存。而多线程的应用则更加广泛，相信在今后的实验中还会多次用到。