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	15 级	专业（方向）	软件工程(移动)
学号	15352133	姓名	黄少豪
电话	13727024545	Email	328730316@qq.com
开始日期	2017.12.2	完成日期	2017.12.3

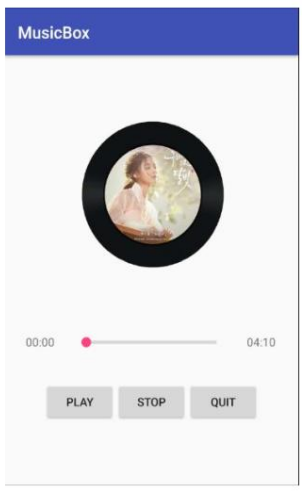
一、 实验题目

服务与多线程--简单音乐播放器

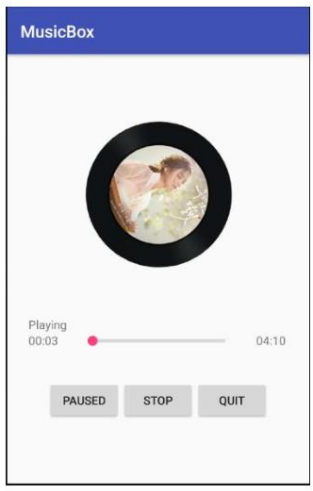
二、 实现内容

实现一个简单的播放器，要求功能有：

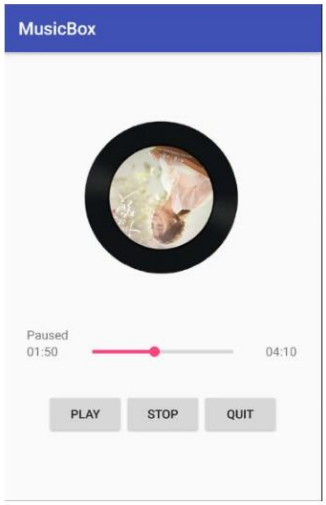
1. 播放、暂停，停止，退出功能；
2. 后台播放功能；
3. 进度条显示播放进度、拖动进度条改变进度功能；
4. 播放时图片旋转，显示当前播放时间功能；



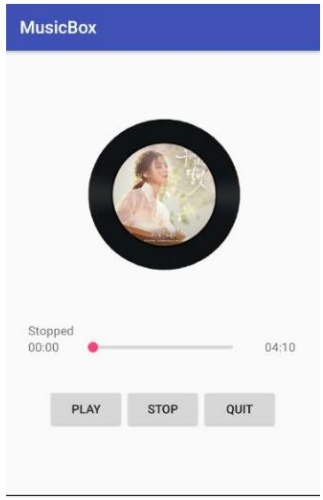
打开程序主页面



开始播放



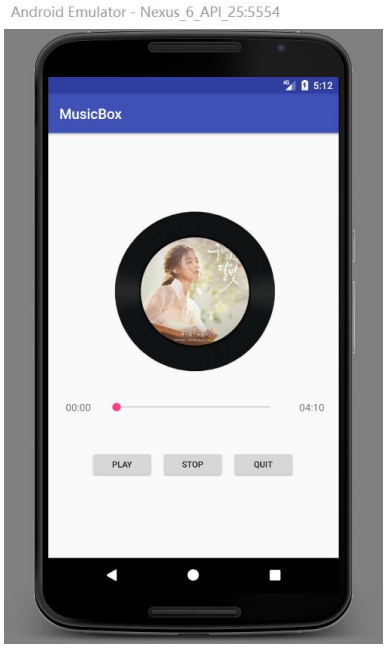
暂停

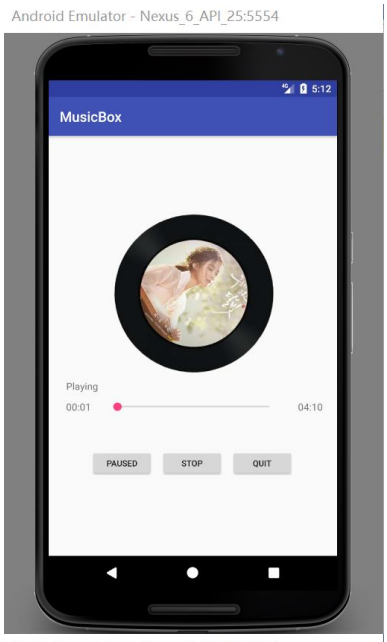


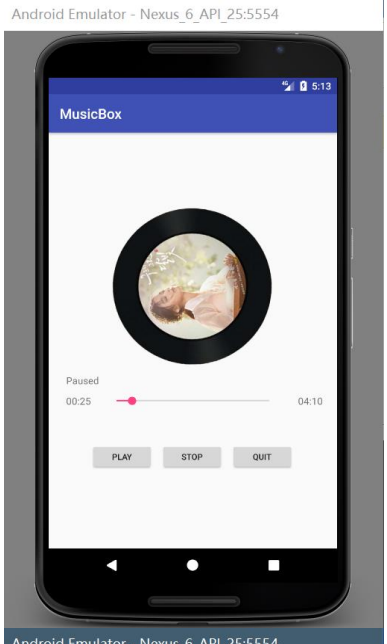
停止

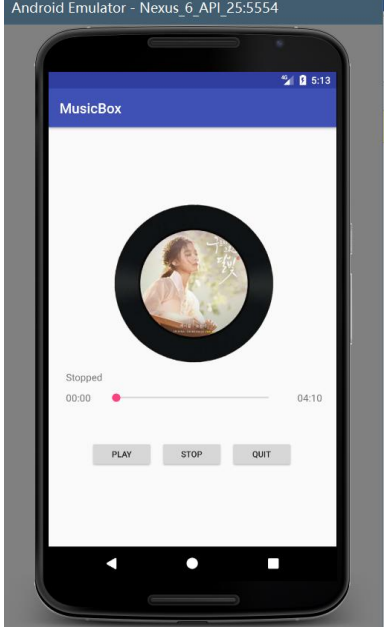
三、 课堂实验结果

（1） 实验截图




Android Emulator - Nexus_6_API_25:5554
5:12
MusicBox
Playing
00:01
04:10
PAUSED
STOP
QUIT


Android Emulator - Nexus_6_API_25:5554
5:13
MusicBox
Paused
00:25
04:10
PLAY
STOP
QUIT


Android Emulator - Nexus_6_API_25:5554
5:13
MusicBox
Stopped
00:00
04:10
PLAY
STOP
QUIT

（2） 实验步骤以及关键代码

1. 定义一个 MusicService 类作为后台播放器

```
public class MusicService extends Service{
    private IBinder mBinder = new MyBinder();
    private MediaPlayer mp = new MediaPlayer();
```

2. 定义 MyBinder 类作为 MainActivity 与 MusicService 的通信工具。OnTransact 函数接收 MainActivity 调用 Transact 函数发来的数据。

```
public class MyBinder extends Binder {
    @Override
    protected boolean onTransact(int code, Parcel data, Parcel reply, int flags) throws RemoteException{
        if (code == 101){   //播放/暂停音乐
            if (mp.isPlaying())
                mp.pause();
            else
                mp.start();
        }
        else if (code == 102){  //停止音乐
            mp.stop();
            try {
                mp.prepare();
                mp.seekTo(0);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
        else if (code == 103)   //退出应用
            mp.release();
         else if (code == 104)   //更新进度条
             reply.writeInt(mp.getCurrentPosition());
         else if (code == 105)   //修改音乐播放进度
             mp.seekTo(data.readInt());
         else{                   //初始化音乐播放器
             try{
                 /*
                 AssetManager am = getAssets();
                 AssetFileDescriptor afd = am.openFd("melt.mp3");
                 mp.setDataSource(afd.getFileDescriptor(),
                         afd.getStartOffset(), afd.getLength());
                 */
                 mp.setDataSource(Environment.getExternalStorageDirectory() + "/melt.mp3");
                 mp.prepare();
                 mp.setLooping(true);
                 reply.writeInt(mp.getDuration());
                 reply.writeInt(mp.getCurrentPosition());
             } catch (Exception e) {
                 e.printStackTrace();
             }
         }
         return super.onTransact(code, data, reply, flags);
    }
}
```

当 code 为 101 时修改播放器的播放与暂停状态；当 code 为 102 时将播放器停止音乐播放器并设置播放器为初始状态；当 code 为 103 时退出音乐，释放播放器；当 code 为 104 时返回音乐播放进度给 MainActivity 使其能更新 UI 进度条；当 code 为 105 时表示 MainActivity 修改了 UI 进度条的进度，此时需要调整音乐播放器使音乐播放进度符合进度条进度；当 code 不为以上任何一种情况，初始化音乐播放器，设置音乐播放器的一些基本信息，比如播放的音乐文件设置，循环播放设置等。

3. MainActivity 动态申请文件访问权限

首先在 AndroidManifest.xml 文件中配置读权限

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

接着在 MainActivity 中动态申请读权限

```
public static void verifyStoragePermissions(Activity activity) {
    try {
        int permission = ActivityCompat.checkSelfPermission(activity, "android.permission.READ_EXTERNAL_STORAGE");
        if (permission != PackageManager.PERMISSION_GRANTED) {
            ActivityCompat.requestPermissions(activity, PERMISSIONS_STORAGE, REQUEST_EXTERNAL_STORAGE);
        }
        else {
            hasPermission = true;
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

当没有获得权限时会弹出对话框供选择是否打开权限，用户选择后此时会调用一个回调函数，代码如下：

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

当用户选择了打开权限，就将 hasPermission 设置为 true，否则就退出应用

4. 将 MainActivity 与 MusicService 通过 ServiceConnection 对象 mConnection 建立连接，实现互相通信

```
mConnection = new ServiceConnection() {
    @Override
    public void onServiceConnected(ComponentName name, IBinder service) {
        mBinder = service;
        try {
            int code = 110;
            Parcel data = Parcel.obtain();
            Parcel reply = Parcel.obtain();
            mBinder.transact(code, data, reply, 0);
            seekBar.setMax(reply.readInt());
            seekBar.setProgress(reply.readInt());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    @Override
    public void onServiceDisconnected(ComponentName name) { mConnection = null; }
};
Intent intent = new Intent(this, MusicService.class);
startService(intent);
bindService(intent, mConnection, Context.BIND_AUTO_CREATE);
```

当连接建立成功，mConnection 会执行 onServiceConnected 函数。mBinder 是一个 IBinder 对象，MainActivity 与 MusicService 通过 mBinder 互相传递数据，其中 code 是指令码，data 存储 MainActivity 传递的数据，reply 存储 MusicService 返回的数据。MainActivity 通过

mBinder 调用 transact 函数将数据传递给 MusicService。

5. 给 seekBar 设置进度条修改监听器

```
seekBar.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
    @Override
    public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {
        if (fromUser){
            try {
                int code = 105;
                Parcel data = Parcel.obtain();
                Parcel reply = Parcel.obtain();
                data.writeInt(progress);
                mBinder.transact(code, data, reply, 0);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }

    @Override
    public void onStartTrackingTouch(SeekBar seekBar) {}

    @Override
    public void onStopTrackingTouch(SeekBar seekBar) {}
});
```

当进度条发生改变且改变来自与用户操作，则传递 code=105 给 MusicService 让其修改播放器的播放进度。

6. 创建一个线程负责执行进度条进度的修改与更新工作

```
Thread mThread = new Thread(){
    @Override
    public void run(){
        while (true) {
            try {
                Thread.sleep(100);
            } catch (Exception e) {
                e.printStackTrace();
            }
            if (mConnection != null && hasPermission == true)
                mHandler.obtainMessage(mark).sendToTarget();
        }
    }
};
mThread.start();
```

因为修改 MainActivity 的 UI 只能是 MainActivity 主线程，这个创建的线程并不能修改进度条进度，所以需要通过 Handler 对象更新进度条

7. 创建一个 Handler 对象负责帮助创建的线程更新主线程 UI

```
final Handler mHandler = new Handler(){
    @Override
    public void handleMessage(Message msg){
        super.handleMessage(msg);
```

```
try {
    int code = 104;
    Parcel data = Parcel.obtain();
    Parcel reply = Parcel.obtain();
    mBinder.transact(code, data, reply, 0);
    int progress = reply.readInt();
    seekBar.setProgress(progress);
    time1.setText(time.format(progress));
} catch (Exception e) {
    e.printStackTrace();
}
```

传递 code=104 给 MusicService 使其返回音乐播放器播放进度，播放进度保存在 reply 数据包中，从 reply 数据包中取出播放器播放进度，更新进度条进度。

（3） 实验遇到困难以及解决思路

1. 一开始不太理解 handler 的实际用途，觉得直接在线程里面更新 UI 就好了，然后报错说不能在线程里面更新主线程的 UI，于是改成在线程调用 handler 修改 UI 就解决问题了。

2. 在 MainActivity 调用 mBinder 初始化 MusicService 的 mediaPlayer 后，音乐文件并没有成功导入 mediaPlayer。后来发现疑似是 mConnection 还没有完成建立 MainActivity 与 MusicService 的连接就调用了 mBinder 传递数据。解决方法是将 MainActivity 通过 mBinder 传递数据给 MusicService 使其初始化 mediaPlayer 的操作写在 mConnection 的 onServiceConnected 函数中。

四、 课后实验结果

通过申请动态权限读取 内置 sd 卡中音乐文件

(具体实现见实验步骤以及关键代码第 3 点)

五、 实验思考及感想

1. 非主线程更新 UI 的操作需要通过 Handler 对象来实现
2. Handler 对象的操作是由主线程完成的。子线程可以进行耗时操作，

更新 UI 的操作通过给 Handler 对象传递消息让主线程通过 Handler 对象完成 UI 更新操作

3. 文件资源的调用还可以通过将文件传入 assets 文件内调用，但这种方式会使 app 占用很大且更新资源也相对不太灵活，还是使用 android 内置的访问系统文件路径来获取文件的方式较好。
4. Activity 与 Service 通过 ServiceConnection 建立连接，并通过 ServiceConnection 的回调函数 onServiceConnected 返回的 IBinder 对象进行数据交流。
5. 重载实现的 onTransact 函数实际是在 transact 函数中被调用的。