

宝贵建议请发送至：wangzhenyang@itcast.cn



黑马程序员

itheima.com

-编程，始于黑马

Android 课程同步笔记

Alpha 0.01 版

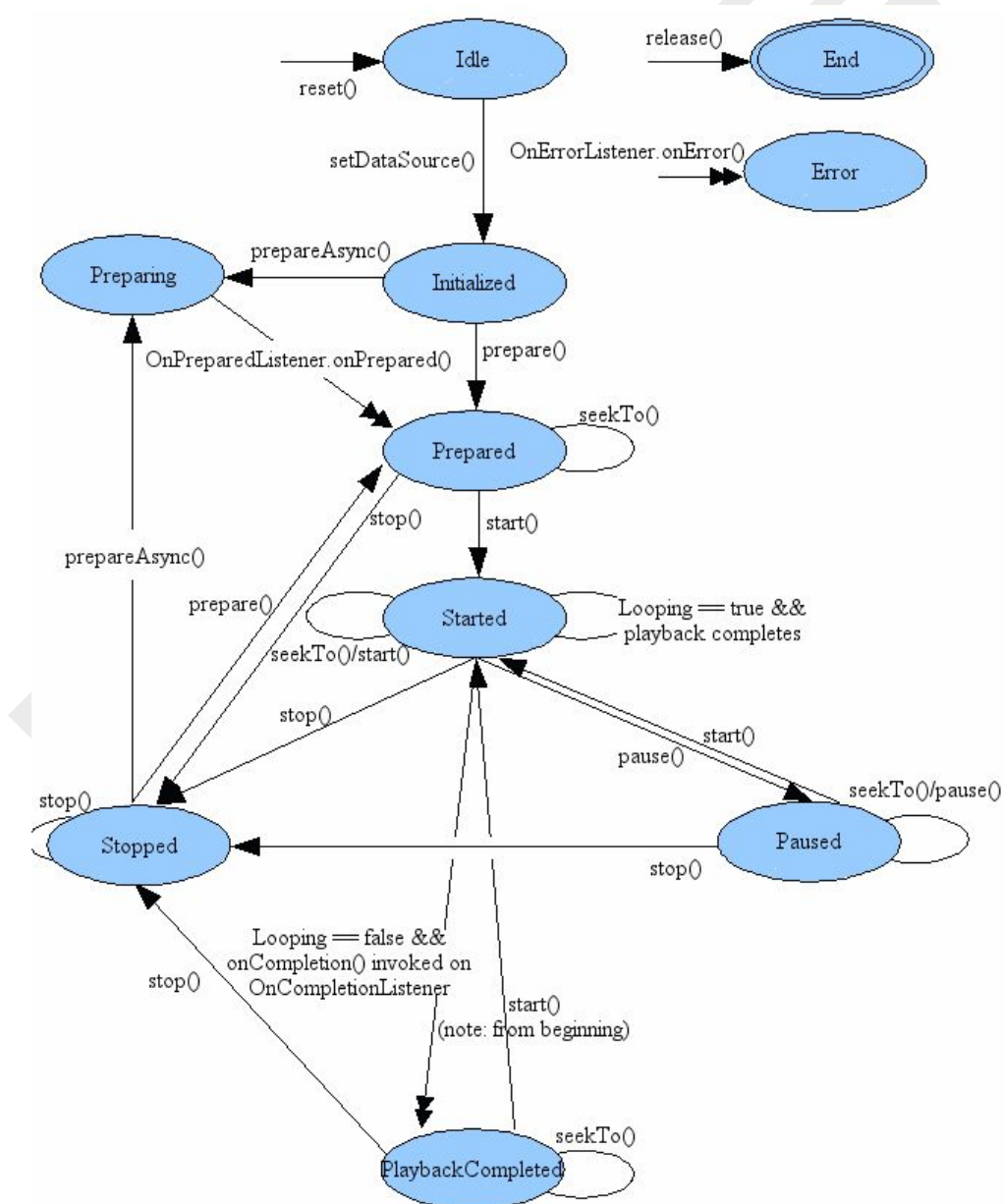
By 阳哥

Android-多媒体&传感器

1. 音频播放 (★★)

1.1 MediaPlayer

MediaPlayer 播放器状态流程如下图：



该播放器同时只能播放一个音乐文件，文件大小并没有限制。

Tips：MediaPlayer 必须严格按照状态图操作，否则就会出现错误，这些错误都是底层抛出，严格按照状态图操作的话一般就不会出问题。

◆ 使用 MediaPlayer 播放音乐的步骤

```
MediaPlayer player = new MediaPlayer();    创建对象

player.reset();                            重置为初始状态

player.setAudioStreamType(AudioManager.STREAM_MUSIC); 声音流类型

player.setDataSource( "/mnt/sdcard/test.mp3" ); 设置音频源

player.prepare();                          准备

player.start();                            开始或恢复播放

player.pause();                            暂停播放

player.start();                            恢复播放

player.stop();                             停止播放

player.release();
```

1.2 SoundPool

SoundPool 和其他声音播放类相比，其特点是可以自行设置声音的品质、音量、播放比率等参数。并且它可以同时管理多个音频流，每个流都有独自的 ID，对某个音频流的管理都是通过 ID 进行的。

◆ 1. SoundPool 最大只能申请 1M 的内存空间，这就意味着我们只能用一些很短的声音片段，而不是用它来播放歌曲或者做游戏背景音乐。

2. SoundPool 提供了 pause 和 stop 方法，但这些方法建议最好不要轻易使用，因为有时候它们可能会使你的程序莫名其妙的终止。有些朋友反映它们不会立即中止播放声音，

而是把缓冲区里的数据播放完才会停下来，也许会多播放一秒钟。

3. SoundPool 的效率问题。其实 SoundPool 的效率在这些播放类中算是很好的了，这可能会影响用户体验。也许这不能管 SoundPool 本身，因为到了性能比较好的 Droid 中这个延迟就可以让人接受了。

◆ 使用 SoundPool 播放音乐的步骤

创建 SoundPool 实例，第一个参数用于指定最大可以加载声音个数

```
SoundPool soundPool=new SoundPool(3,AudioManager.STREAM_MUSIC, 0);
```

加载不同的声音文件，生成各自的声音 id

```
int shoot1ID = soundPool.load(this, R.raw.shoot1, 1);
```

```
int shoot2ID = soundPool.load(this, R.raw.shoot2, 1);
```

```
int shoot3ID = soundPool.load(this, R.raw.shoot3, 1);
```

根据 load 方法获取的 id 播放对应的声音

```
soundPool.play(shoot1ID, 1, 1, 1, 0, 1);
```

```
soundPool.play(shoot2ID, 1, 1, 1, 0, 1);
```

```
soundPool.play(shoot3ID, 1, 1, 1, 0, 1);
```

1.3 使用两种方式播放音乐

该案例比较简单，布局中只需两个按钮，一个用于播放 MediaPlayer 一个用于播放 SoundPool。我们需要将事先准备好的声音文件放到 res/raw 目录下，其中 raw 目录需要我们新创建。

代码清单如下：

```
public class MainActivity extends Activity {
    //声明 MediaPlayer 和 SoundPool 对象
    private MediaPlayer mediaPlayer;
    private SoundPool soundPool;
    private int soundID;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //创建 MediaPlayer 对象
        mediaPlayer = new MediaPlayer();
        try {
            //设置音乐文件路径
            mediaPlayer.setDataSource(Environment.getExternalStorageDirectory()+"/bgmusic.mp3");
            //准备
            mediaPlayer.prepare();
        } catch (Exception e) {
            e.printStackTrace();
        }
        //初始化 SoundPool
        soundPool = new SoundPool(1, AudioManager.STREAM_MUSIC,1);
        //加载 SoundPool 音乐文件
        soundID = soundPool.load(this, R.raw.shoot1, 1);
    }

    public void soundPoolPlay(View view) {
        //播放 SoundPool 音乐
        soundPool.play(soundID, 1, 1, 0, 0, 1);
    }

    public void mediaPlayerPlay(View view) {
        //判断当前音乐是否在播放
        boolean isPlaying = mediaPlayer.isPlaying();
        if (!isPlaying) {
            //播放音乐
            mediaPlayer.start();
        } else {
            Toast.makeText(this, "音乐正在播放中", 0).show();
        }
    }
}
```

1.4 案例-音乐播放器

需求：制作一个播放器，能够播放/暂停/停止/重播音乐文件，并且添加一个 SeekBar, 音乐播放时，SeekBar 的滚动条也会变化，拖动 SeekBar，可更改声音播放的进度。

注意：要实现此音乐播放器，必须严格按照上方的 MediaPlayer 状态流程图进行控制，否则非常容易使应用程序出错挂掉。

1

新创建一个 Android 工程《音乐播放器》

2

修改并使用默认布局文件，布局文件清单如下：

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical"
tools:context=".MainActivity" >

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:text="音乐播放器" />

    <EditText
        android:id="@+id/et_path"
        android:text="/mnt/sdcard/bgmusic.mp3"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="请输入音乐路径" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal" >

        <Button
            android:layout_width="0dp"
```

```
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:onClick="play"
        android:text="播放" />

<Button
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:onClick="pause"
    android:text="暂停" />

<Button
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:onClick="stop"
    android:text="停止" />

<Button
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:onClick="replay"
    android:text="重播" />
</LinearLayout>

<SeekBar
    android:id="@+id/sb"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />

<TextView
    android:id="@+id/tv_time"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="right"
    android:textColor="#ff0000"
    android:textSize="24sp" />

</LinearLayout>
```

3

使用并编写默认的 MainActivity 类，在该类中实现核心业务逻辑。代码清单如下：

```
public class MainActivity extends Activity implements
OnSeekBarChangeListener {
    private EditText et_path;
    private SeekBar sb;
    private TextView tv_time;
    private MediaPlayer player;
    private int duration;
    private boolean isUpdateBar;
    // 播放器的几个状态
    private static final int PLAYING = 1; // 播放状态
    private static final int PAUSING = 2; // 暂停状态
    private static final int STOPPING = 3; // 停止状态
    private int CURRENT = 0; // 当前状态

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //初始化界面控件
        et_path = (EditText) findViewById(R.id.et_path);
        sb = (SeekBar) findViewById(R.id.sb);
        //给 SeekBar 绑定滑动事件
        sb.setOnSeekBarChangeListener(this);
        tv_time = (TextView) findViewById(R.id.tv_time);
    }
    //播放音乐
    public void play(View view) {
        if (player!=null) {
            if (CURRENT==PLAYING) {
                return;
            }else if (CURRENT==PAUSING) {
                player.start();
                CURRENT = PLAYING;
                updateSeekBar();
                return;
            }else if (CURRENT==STOPPING) {
                player.reset();
                player.release();
            }
        }
    }
}
```



```
}  
    try {  
        //创建一个播放器对象  
        player = new MediaPlayer();  
        //获取音乐路径  
        String path = et_path.getText().toString();  
        //给播放器设置音乐路径  
        player.setDataSource(path);  
        //设置音乐格式  
        player.setAudioStreamType(AudioManager.STREAM_MUSIC);  
        //准备  
        player.prepare();  
        //获取音乐最大长度（毫秒单位）  
        duration = player.getDuration();  
        //给SeekBar设置最大值  
        sb.setMax(duration);  
        //格式化输出音乐长度  
        String lastString = formatTime(duration);  
        tv_time.setText("00:00/" + lastString);  
        //音乐开始播放  
        player.start();  
        //更新SeekBar  
        updateSeekBar();  
    } catch (Exception e) {  
        e.printStackTrace();  
        Toast.makeText(this, "音乐播放失败"+e, 0).show();  
    }  
}  
//暂停  
public void pause(View view) {  
    if (player != null && CURRENT==PLAYING) {  
        player.pause();  
        stopSeekBar();  
        CURRENT = PAUSING;  
    }  
}  
//停止播放  
public void stop(View view) {  
    if (player != null) {  
        if (CURRENT == PLAYING || CURRENT == PAUSING) {  
            player.stop();  
        }  
    }  
}
```

```

        stopSeekBar();
        sb.setProgress(0);
        tv_time.setText("00:00/"+formatTime(duration));
        CURRENT = STOPPING;
    }
}
//重新播放
public void replay(View view) {
    if (player != null) {
        stopSeekBar();
        CURRENT = STOPPING;
        play(view);
    }
}
//停止 SeekBar 更新
private void stopSeekBar() {
    isUpdateBar = false;
}
//更新 SeekBar
private void updateSeekBar() {
    isUpdateBar = true;
    new Thread(new Runnable() {

        @Override
        public void run() {
            while (isUpdateBar) {
                //每秒更新一次
                SystemClock.sleep(1000);
                if (player != null && CURRENT==PLAYING) {
                    sb.setProgress(player.getCurrentPosition());
                    //在主线程中运行如下代码更新 EditText
                    runOnUiThread(new Runnable() {

                        @Override
                        public void run() {
                            String current =
formatTime(player.getCurrentPosition());
                            String durationString =
formatTime(duration);

```

```
tv_time.setText(current + "/" + durationString);
    }
    });
    }
    }
    }
    }).start();
}

@Override
public void onProgressChanged(SeekBar seekBar, int progress,
boolean fromUser) {
    if (player != null) {
        player.seekTo(progress);
    }
}

//开始拖动 SeekBar 时停止更新 SeekBar
public void onStartTrackingTouch(SeekBar seekBar) {
    stopSeekBar();
}

//停止拖动 SeekBar 的时候将音乐定位到相应位置
public void onStopTrackingTouch(SeekBar seekBar) {
    if (player != null) {
        player.seekTo(seekBar.getProgress());
        updateSeekBar();
    }
}

//工具函数 格式化播放时间
private String formatTime(int current) {
    int second = current / 1000;
    int minute = second / 60;
    second = second - minute * 60;
    StringBuilder sb = new StringBuilder();
    sb.append(minute > 10 ? minute + " : " + minute);
    sb.append(":");
    sb.append(second > 10 ? second : "0" + second);
    return sb.toString();
}
}
```

4

运行上面的工程，效果图如下：



2. 视频播放 (★★)

2.1 使用 MediaPlayer+SurfaceView 播放视频文件

Tips：在这里视频播放依然通过 MediaPlayer 类，为了方便演示，我们直接使用 1.4

章节中的创建的工程，只需在布局文件添加 SurfaceView 控件即可。

```
<SurfaceView
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/sv"
/>
```

我们直接在 1.4 章节的代码基础上进行修改，修改 MainActivity 类，这里只给出不同的代

码片段：

◆ 添加如下变量的声明

```
private SurfaceView sv;
private SurfaceHolder holder;
```

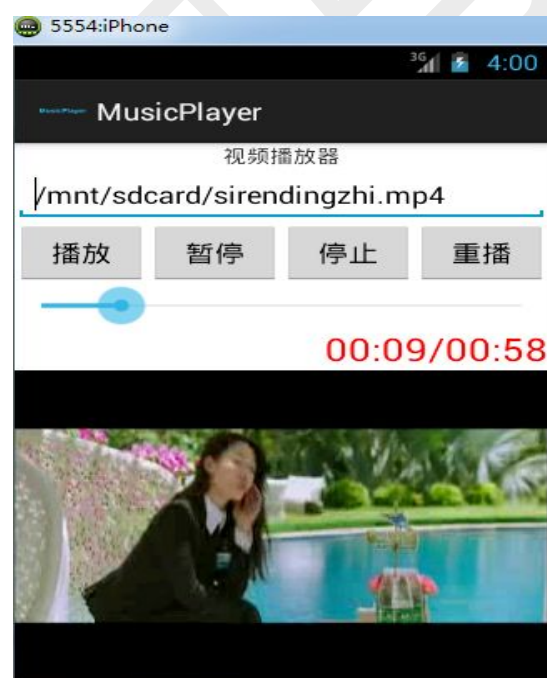
◆ 在 play () 方法中添加如下代码

```
//设置输出画面
player.setDisplay(holder);
```

◆ 获取 SurfaceView 对象，并设置缓存方式

```
sv = (SurfaceView) findViewById(R.id.sv);
holder = sv.getHolder();
/**
 * SurfaceView 内部采用双缓冲区
 * 设置 SurfaceView 不维护自己的缓存区，使用屏幕的渲染引擎将内容推
 * 送到用户面前
 */
holder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
```

运行该工程，效果图如下：



2.2 使用 VideoView 控件显示视频

VideoView 跟 MediaPlayer 相比播放视频步骤要简单的多，因此 VideoView 自己提供了播放，暂停、快进、快退、进度条等方法。使用起来要方便的很多。

步骤：1、设置布局文件，布局文件比较简单，因此这里只给你 VideoView 标签。

```
<VideoView
    android:id="@+id/vv"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

2、设置 VideoView 的播放文件路径和媒体控制器,调用 start 方法即可播放媒体文件。

```
//实例化 VideoView 对象
vv = (VideoView) findViewById(R.id.vv);
//从界面获取播放路径
et_path = (EditText) findViewById(R.id.et_path);
//给 VideoView 设置视频路径
vv.setVideoPath(et_path.getText().toString());
//设置 VideoView 控制器，我们当前类实现了 MediaPlayerControl 接口
vv.setMediaController(new MediaController(this));
//开始播放
vv.start();
//设置当前播放器窗口设置为焦点
vv.requestFocus();
```

3、覆写 MediaPlayerControl 接口中的抽象方法

```
@Override
public void start() {
}
@Override
public void pause() {
}

@Override
public int getDuration() {
    return 0;
}
@Override
```

```
public int getCurrentPosition() {  
    return 0;  
}  
@Override  
public void seekTo(int pos) {  
}  
@Override  
public boolean isPlaying() {  
    return false;  
}  
@Override  
public int getBufferPercentage() {  
    return 0;  
}  
@Override  
public boolean canPause() {  
    return false;  
}  
@Override  
public boolean canSeekBackward() {  
    return false;  
}  
@Override  
public boolean canSeekForward() {  
    return false;  
}  
@Override  
public int getAudioSessionId() {  
    return 0;  
}
```

Tips：上面的方法都是回调方法，我们可以在这些方法里面实现我们的业务逻辑。但是当

我们给 VideoView 设置 setMediaController 后，控制器才会出现。

4、运行上面代码，效果图如下：



3. 传感器 (★★)

3.1 Android 中常见的传感器

Android 手机中内置了很多传感器，其主要类型有：方向、加速度(重力)、光线、磁场、距离(临近性)、温度等。

◆ 方向传感器	Sensor.TYPE_ORIENTATION
◆ 加速度(重力)传感器	Sensor.TYPE_ACCELEROMETER
◆ 光线传感器	Sensor.TYPE_LIGHT
◆ 磁场传感器	Sensor.TYPE_MAGNETIC_FIELD



距离(临近性)传感器

Sensor.TYPE_PROXIMITY



温度传感器

Sensor.TYPE_TEMPERATURE

3.2 传感器的使用

获取传感器管理器 `SensorManager`:

```
//获取传感器管理器  
SensorManager sm = (SensorManager) getSystemService(SENSOR_SERVICE);
```

通过传感器管理器对象获得指定类型的传感器：

```
//获取指定传感器对象，获取系统默认的重力加速度传感器  
Sensor sensor = sm.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
```

通过传感器管理器对象获得手机中所有的传感器：

```
//获取手机支持的所有传感器  
List<Sensor> sensorList = sm.getSensorList(Sensor.TYPE_ALL);  
for(int i=0;i<sensorList.size();i++){  
    Sensor sensor = sensorList.get(i);  
    //获取传感器名称  
    String name = sensor.getName();  
    //获取传感器厂商  
    String vendor = sensor.getVendor();  
    //获取传感器版本号  
    int version = sensor.getVersion();  
}
```

使用传感器管理器对象注册传感器来使一个传感器工作：

```
/*  
 * 注册一个传感器  
 * 第二个参数 Sensor 是具体某个传感器对象  
 * 第三个参数是设定传感器采样频率  
 */  
sm.registerListener(new SensorEventListener() {  
    //当传感器值改变时触发该函数
```

```
@Override
    public void onSensorChanged(SensorEvent event) {
        Toast.makeText(MainActivity.this, ""+event.accuracy,
0).show();
    }
    //当传感器精确度更改是触发该函数
    @Override
    public void onAccuracyChanged(Sensor sensor, int accuracy)
{
        Toast.makeText(MainActivity.this, ""+accuracy,
1).show();
    }
}, sensor, SensorManager.SENSOR_DELAY_NORMAL);
```

Tips:

listener : 传感器事件监听器

sensor : 要被注册的传感器对象

rate : 采样率,分为最快、游戏、普通、用户界面几种当应用程序请求特定的采样率时,其实只是对传感器子系统的一个建议,不保证特定的采样率可用。

采样率的四种类型详解:

最快: SensorManager.SENSOR_DELAY_FASTEST

最低延迟,一般不是特别敏感的处理不推荐使用,该种模式可能造成手机电力大量消耗,由于传递的为原始数据,算法不处理好将会影响游戏逻辑和 UI 的性能

游戏: SensorManager.SENSOR_DELAY_GAME

游戏延迟,一般绝大多数的实时性较高的游戏都使用该级别

普通: SensorManager.SENSOR_DELAY_NORMAL

标准延迟,对于一般的益智类或 EASY 级别的游戏可以使用,但过低的采样率可能对一些赛车类游戏有跳帧现象

用户界面：SensorManager.SENSOR_DELAY_UI

一般对于屏幕方向自动旋转使用，相对节省电能和逻辑处理，一般游戏开发中我们不使用

3.3 案例-公交防盗

需求：当手机从衣服兜里被掏出时，手机响铃报警。

原理：使用距离传感器，当距离从 0 变为 1 时，使用 MediaPlayer 播放声音文件报警

步骤：

1. 设置布局文件：添加一个报警按钮开关，用于开启/停止报警功能
2. 声明成员变量

```
// 声明 Button 变量
private Button bt;
// 声明传感器管理器
private SensorManager sm;
// 声明 MediaPlayer
private MediaPlayer player;
// 声明一个传感器对象
private Sensor sensor;
// 标记当前防盗功能是否开启
private boolean isOpen = false;
```

3. 使当前类继承 SensorEventListener 并覆写抽象方法

```
@Override
public void onSensorChanged(SensorEvent event) {
    // 获取距离传感器的值
    float value = event.values[0];
    if (value > 0) {
        try {
            // 如果距离大于 0 则播放音乐
            player.start();
        } catch (Exception e) {
```

```

        e.printStackTrace();
    }
} else {
    // 如果距离等于 0 暂停音乐
    player.pause();
}
}
@Override
public void onAccuracyChanged(Sensor sensor, int accuracy) {
}

```

4. 实现 Button 的按钮事件，点击开启或者关闭音乐

```

public void click(View view) {
    if (isOpen) { // 如果开启
        // 暂停音乐
        player.pause();
        // 使按钮显示提示文字
        bt.setText("防盗已经关闭");
        // 取消传感器监听
        sm.unregisterListener(this);
        isOpen = false;
    } else { // 如果没有开启
        // 播放音乐
        player.start();
        bt.setText("防盗已经开启");
        // 注册传感器监听
        sm.registerListener(this, sensor,
            SensorManager.SENSOR_DELAY_NORMAL);
        isOpen = true;
    }
}
}

```

5. 覆写 Activity 的 onCreate 方法，在该方法里面实现 MediaPlayer 的初始化以及传感器的监听

```

protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
}

```


```
// 初始化变量
bt = (Button) findViewById(R.id.bt);
sm = (SensorManager) getSystemService(SENSOR_SERVICE);
// 获取距离传感器
sensor = sm.getDefaultSensor(Sensor.TYPE_PROXIMITY);
// 创建 MediaPlayer 对象
player = MediaPlayer.create(this, R.raw.apple);
// 给 MediaPlayer 设置播放完事件监听
player.setOnCompletionListener(new OnCompletionListener() {
    @Override
    public void onCompletion(MediaPlayer mp) {
        // 播放完后重新播放
        mp.start();
    }
});
}
```

4. 摄像头 (★★)

4.1 调用系统摄像头实现照相和摄像功能

调用系统摄像头进行拍照和摄像无需添加权限，直接调用即可。只需知道系统摄像头的 action 和 category 就可以调用系统摄像头。

步骤：

1. 打开 Android 源码，查看“\packages\apps\” 文件目录下的 Camera 应用，即系统摄像头的应用程序。打开其清单文件，查看其 Activity 的 action 和 category 信息。
2.  Camera 类的 action 和 category

```
<intent-filter>
    <action android:name="android.media.action.IMAGE_CAPTURE" />
    <category android:name="android.intent.category.DEFAULT" />
</intent-filter>
```

◆ VideoCamera 类的 action 和 category

```
<intent-filter>
    <action android:name="android.media.action.VIDEO_CAMERA" />
    <category android:name="android.intent.category.DEFAULT" />
</intent-filter>
```

3. 已知调用系统摄像头拍照和摄像功能对应的 action 和 category 信息,采用隐式调用的方式调用 Activity。

由于希望在调用拍照或摄像功能后回到当前应用的界面,且得知拍照或摄像的结果如何,是否成功,所以在开启 Activity 时不能使用 startActivity 方法,而是使用 startActivityForResult 方法开启 Activity,并重写 onActivityResult 方法处理回传的数据。

Tips：布局文件比较简单,界面只有两个按钮,一个用于打开照相机,一个用于打开摄像机。这里只给出代码清单。

◆ 拍照功能

```
public void take(View view){
    //创建一个 Intent 对象
    Intent intent = new Intent();
    //设置 Action
    intent.setAction("android.media.action.IMAGE_CAPTURE");
    //创建一个文件
    File file = new
    File(Environment.getExternalStorageDirectory().getAbsolutePath(), "
    my.jpg");
    //创建 uri 对象
    Uri uri = Uri.fromFile(file);
```

```
//设置图片的输出路径
intent.putExtra(MediaStore.EXTRA_OUTPUT, uri);
//开启 Activity
startActivityForResult(intent, 100);
}
```

◆ 摄像功能（摄像功能跟拍照功能比仅仅是 action 不同而已）

```
public void video(View view){
    Intent intent = new Intent();
    intent.setAction("android.media.action.VIDEO_CAPTURE");
    File file = new
File(Environment.getExternalStorageDirectory().getAbsolutePath(), "
myVedio.mp4");
    Uri uri = Uri.fromFile(file);
    intent.putExtra(MediaStore.EXTRA_OUTPUT, uri);
    startActivityForResult(intent, 101);
}
```

4.2 编码实现照相功能

使用 Camera+SurfaceView 控件可实现拍照功能。

步骤：

1. 设置布局文件：一个 SurfaceView 和一个拍照按钮
2. 在清单文件中添加 2 个权限信息

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

3. 声明成员变量

```
private Camera camera;//照相机对象
private SurfaceHolder holder;//SurfaceView 的辅助类
```

4. 获取 SurfaceView 控件的实例，通过其辅助类 SurfaceHolder 对象添加

CallBack 接口的实现

```
//获取预览画面的 SurfaceView 控件
SurfaceView surfaceView = (SurfaceView) findViewById(R.id.sv);
//得到预览画面的辅助类
holder = surfaceView.getHolder();
//添加回调方法
holder.addCallback(new Callback() {

    @Override
    public void surfaceCreated(SurfaceHolder holder) {
        try {
            //打开摄像头
            camera = Camera.open();
            //设置预览显示的位置
            camera.setPreviewDisplay(holder);
            //开启预览
            camera.startPreview();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    @Override
    public void surfaceDestroyed(SurfaceHolder holder) {
        //停止预览界面
        camera.stopPreview();
        //释放摄像头
        camera.release();
    }

    @Override
    public void surfaceChanged(SurfaceHolder holder, int format,
        int width, int height) {
    }
}
```

5. 添加点击拍照按钮的点击事件：设置摄像头自动对焦，对焦完成后保存图片并重新预览


```
//拍照
public void takeImage(View view) {
    if (camera == null) {
        Toast.makeText(this, "照相机还没打开。",
            Toast.LENGTH_SHORT).show();
        return;
    }
    //点击拍照按钮，摄像头自动对焦，对焦完成后拍照并保存
    camera.autoFocus(new AutoFocusCallback() {

        @Override
        public void onAutoFocus(boolean success, Camera camera) {
            //对焦完成，拍照并保存
            camera.takePicture(null, null, new PictureCallback() {
                @Override
                public void onPictureTaken(byte[] data, Camera
camera) {

                    //设置照片保存路径
                    String path =
Environment.getExternalStorageDirectory().getAbsolutePath() + "/" +
UUID.randomUUID().toString() + ".jpg";
                    try {
                        FileOutputStream outputStream = new
FileOutputStream(path);
                        //将照片字节数组写到文件中
                        outputStream.write(data);
                        outputStream.close();
                        //重新开启预览
                        camera.startPreview();
                    } catch (Exception e) {
                        e.printStackTrace();
                    }
                }
            });
        }
    });
}
```

4.3 编码实现摄像功能

使用 Camera+MediaRecorder + SurfaceView 控件可实现录制视频的功能。

所需权限：

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

步骤：

1. 设置布局文件：一个 SurfaceView+一个开始录制按钮+一个停止录制按钮
2. 在清单文件中添加权限信息：摄像头+录音+写 SD 卡
3. 声明要使用的成员变量

```
private Camera camera;
private SurfaceHolder holder;
private MediaRecorder recorder;
```

4. 获取 SurfaceHolder，添加回调方法

```
//获取 SurfaceView 控件实例
SurfaceView sv = (SurfaceView) findViewById(R.id.sv);
//获取 SurfaceHolder 对象
holder = sv.getHolder();
//添加回调对象
holder.addCallback(this);
```

5. 重写 Callback 接口的 surfaceCreated 和 surfaceDestroyed 方法 :SurfaceView

创建时初始化摄像头,销毁时释放摄像头资源

```
@Override
public void surfaceCreated(SurfaceHolder holder) {
    //打开摄像头
    camera = Camera.open();
    try {
        //给摄像头设置预览对象
```

```
        camera.setPreviewDisplay(holder);
        //开启预览
        camera.startPreview();
    } catch (Exception e) {
        e.printStackTrace();
    }
}

@Override
public void surfaceChanged(SurfaceHolder holder, int format, int
width, int height) {

}

@Override
public void surfaceDestroyed(SurfaceHolder holder) {
    //停止预览
    camera.stopPreview();
    //释放资源
    camera.release();
}
```

6. 添加点击开始录制按钮时的逻辑

```
//开始录制视频
public void start(View view){
    //停止预览
    camera.stopPreview();
    //解锁摄像头
    camera.unlock();
    //初始化一个 MediaRecorder 对象
    recorder = new MediaRecorder();
    //给 recorder 设置摄像头
    recorder.setCamera(camera);
    //设置音频源
    recorder.setAudioSource(AudioSource.CAMCORDER);
    //设置视频源
    recorder.setVideoSource(VideoSource.CAMERA);
    //设置录像质量等参数
    CamcorderProfile profile =
    CamcorderProfile.get(CamcorderProfile.QUALITY_480P);
    recorder.setProfile(profile );
}
```

```
//开始录制视频
public void start(View view){
    //停止预览
    camera.stopPreview();
    //解锁摄像头
    camera.unlock();
    //初始化一个 MediaRecorder 对象
    recorder = new MediaRecorder();
    //给 recorder 设置摄像头
    recorder.setCamera(camera);
    //设置音频源
    recorder.setAudioSource(AudioSource.CAMCORDER);
    //设置视频源
    recorder.setVideoSource(VideoSource.CAMERA);
    //设置录像质量等参数
    CamcorderProfile profile =
    CamcorderProfile.get(CamcorderProfile.QUALITY_480P);
    recorder.setProfile(profile );
    //设置录像输出路径

    recorder.setOutputFile(Environment.getExternalStorageDirectory(
    ).getAbsolutePath()+"/text.mp4");
    //设置预览显示对象
    recorder.setPreviewDisplay(holder.getSurface());
    try {
        //准备
        recorder.prepare();
    } catch (Exception e) {
        e.printStackTrace();
    }
    //开始录像
    recorder.start();
}
```

7. 添加点击停止录制时的逻辑

```
public void stop(View view){
    //停止录像
    recorder.stop();
    //重置录像
    recorder.reset();
    //释放资源
```

```
recorder.release();  
//锁定摄像头  
camera.lock();  
//开启预览  
camera.startPreview();  
}
```

至此，本文档完！

2014 年 12 月 24 日星期三 22:30:43

北京市海淀区东北旺中路东馨园