Android Sensor 实验

一、 实验目的

1、 了解 Android 手机常见传感器
2、 理解 Android 手机常见传感器的工作原理
3、 掌握 Android 赏见传感器的编程使用

二、 实验条件

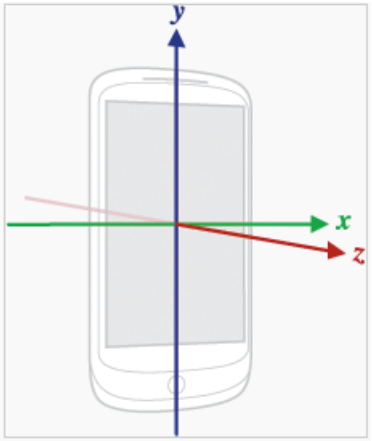
- ✓ IBM-PC 兼容机
- ✓ Windows、Ubuntu11.04 或其他兼容的 Linux 操作系统
- ✓ JDK（建议安装 JDK8 及其以上版本）、Android Studio 或 Eclipse with ADT
- ✓ INTEL ATOM 平板

三、 实验原理

Android 平台支持的所有运动传感器:

传感器	传感器事件数据	说明	测量单位
TYPE_ACCELEROMETER	SensorEvent.values[0]	沿 x 轴的加速度（包括重力）。	m/s2
	SensorEvent.values[1]	沿 y 轴的加速度（包括重力）。	
	SensorEvent.values[2]	沿 z 轴的加速度（包括重力）。	
TYPE_GRAVITY	SensorEvent.values[0]	沿 x 轴的重力加速度。	m/s2
	SensorEvent.values[1]	沿 y 轴的重力加速度。	
	SensorEvent.values[2]	沿 z 轴的重力加速度。	
TYPE_GYROSCOPE	SensorEvent.values[0]	围绕 x 轴的旋转角速度。	rad/s
	SensorEvent.values[1]	围绕 y 轴的旋转角速度。	
	SensorEvent.values[2]	围绕 z 轴的旋转角速度。	
TYPE_LINEAR_ACCELERATION	SensorEvent.values[0]]	沿 x 轴的加速度（不包括重力）。	m/s2
	SensorEvent.values[1]	沿 y 轴的加速度（不包括重力）。	
	SensorEvent.values[2]	沿 z 轴的加速度（不包括重力）。	
TYPE_ROTATION_VECTOR	SensorEvent.values[0]]	旋转向量沿 x 轴的部分（x * sin(θ/2)）。	无无
	SensorEvent.values[1]	旋转向量沿 y 轴的部分（y * sin(θ/2)）。	
	SensorEvent.values[2]]	旋转向量沿 z 轴的部分（z * sin(θ/2)）。	
	SensorEvent.values[3]]	旋转向量的数值部分（(cos(θ/2)）1。	

传感器的坐标系:



以设备为参照物的坐标系

Android 传感器框架是 android.hardware 包的一部分，包含了以下类和接口：

SensorManager

你可以用这个类来创建传感器设备的一个实例。这个类提供了多个方法，用于访问及获取传感器列表、注册及注销传感器事件侦听器、读取方位信息等。

该类还提供了众多的传感器常量，用于报告传感器精度、设置数据采样率和校准传感器。

```
private  SensorManager  mSensorManager;
```

Sensor

你可以用这个类来创建某个传感器的实例。该类提供了很多方法，使你能确定传感器的性能。

```
private  Sensor  mSensor;

mSensor  =  mSensorManager.getDefaultSensor(Sensor.TYPE_GYROSCOPE);
```

SensorEvent

系统用该类来创建一个传感器事件对象，用于提供传感器事件的相关信息。这些信息包括：传感器原始数据、生成本事件的传感器类型、数据的精度、事件的时间戳。

SensorEventListener

你可以用该类来创建两个回调方法，用于传感器数值改变时或传感器精度变化时接收通知（传感器事件）。

识别传感器及其性能:

Android 的传感器框架提供了众多的方法，使你很容易就能在运行时检测出当前设备可用的传感器。 API 也提供了很多检测每个传感器性能的方法，比如量程、分辨率、能耗。

要识别设备上的传感器，你首先需要获取一个传感器设备的引用。你可以通过调用 getSystemService()，并传入 SENSOR_SERVICE 参数，来创建一个 SensorManager。例如：

```
private SensorManager mSensorManager;
...
mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
```

监听传感器事件:

要监控传感器的原始数据，你需要实现 SensorEventListener 接口的 onAccuracyChanged() 和 onSensorChanged() 回调方法。只要发生以下事件，Android 系统就会调用这两个方法：

传感器精度发生变化

在这种情况下，系统会调用 onAccuracyChanged()方法，并传给你一个发生变化的 Sensor 对象的引用和新的传感器精度值。精度用以下四种状态常量之一来表示：SENSOR_STATUS_ACCURACY_LOW、SENSOR_STATUS_ACCURACY_MEDIUM、SENSOR_STATUS_ACCURACY_HIGH、 和 SENSOR_STATUS_UNRELIABLE。

传感器报送一个新数据

这种情况下，系统会调用 onSensorChanged() 方法，并传给你一个 SensorEvent 对象。 SensorEvent 对象中包含了新数据的相关信息，包括：数据精度、生成数据的传感器、生成数据的时间戳、传感器采到的新数据。

调用 registerListener() 时指定了缺省的数据延时(SENSOR_DELAY_NORMAL)。

数据延时（或采样率）控制着由 onSensorChanged() 发送给应用的传感器事件的触发间隔。缺省的数据延迟是 200,000 微秒，适于监测典型的屏幕方向变动.你可以把数据延时指定为其它值，比如 SENSOR_DELAY_GAME （20,000 微秒）、SENSOR_DELAY_UI （60,000 微秒）或 SENSOR_DELAY_FASTEST （0 微秒）。Android 3.0 (API Level 11) 开始，你还可以直接指定延时值（微秒数）.

请确保为你的应用或使用场景选择了合适的发送频率。传感器能够以很高的频率发送数据。 请保证系统有能力发送其它数据，不要无谓浪费系统资源和消耗电池电量。

使用了 onResume() 和 onPause() 回调方法来注册和注销传感器事件侦听器。最佳方案就是，你应该保证在不用时及时关闭传感器，特别当你的 activity 被暂停时。 不然，因为某些传感器的能耗很大，会快速消耗电池电量，可能会在几个小时内将电池耗尽。

设置权限:

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

Layout 布局设置为线性布局:

```
<?xml version="1.0" encoding="utf-8"?>
  <LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">

    <TextView
        android:id="@+id/text_Accelerometer"
        android:layout_width="fill_parent"
```

```
        android:layout_height="wrap_content"
        android:text="Accelerometer:disable" />
    <TextView
        android:id="@+id/text_Gyroscope"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Gyroscope:disable" />
</LinearLayout>
```

示例代码：

MainActivity:

```
package com.example.test;
import android.app.Activity;
import android.content.Context;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.os.Bundle;
import android.view.Menu;
import android.widget.TextView;
public class MainActivity extends Activity {
    private SensorManager mSensorManager;
    private TextView mtextaccelerometer;
    private TextView mtextGyroscope;
    private SensorEventListener mEventListenerAccelerometer;
    private SensorEventListener mEventListenerGyroscope;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
        mtextaccelerometer=(TextView)findViewById(R.id.text_Accelerometer);
        mtextGyroscope=(TextView)findViewById(R.id.text_Gyroscope);
        initListeners();
    }
    private void initListeners(){
        mEventListenerAccelerometer=new SensorEventListener() {
            @Override
            public void onSensorChanged(SensorEvent event) {
                // TODO Auto-generated method stub
                mtextaccelerometer.setText("Accelerometer:    "    +    "\n"+"X    轴    :
"+event.values[0]+
                        "\n"+ "y 轴：" + event.values[1] + "\n " +"z 轴："+ event.values[2]);
```

```
            }
            @Override
            public void onAccuracyChanged(Sensor sensor, int accuracy) {
                // TODO Auto-generated method stub
            }
        };
        mEventListenerGyroscope=new SensorEventListener() {
            @Override
            public void onSensorChanged(SensorEvent event) {
                // TODO Auto-generated method stub
                float[] values=event.values;
                mtextGyroscope.setText("Gyroscope:" + "\n"+"X 轴："+values[0]+
                        "\n"+ "y 轴：" + values[1] + "\n " +"z 轴："+ values[2]);
            }
            @Override
            public void onAccuracyChanged(Sensor sensor, int accuracy) {
                // TODO Auto-generated method stub
            }
        };
    }
    @Override
    protected void onResume() {
        super.onResume();
        mSensorManager.registerListener(mEventListenerAccelerometer,
                mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER),
                SensorManager.SENSOR_DELAY_NORMAL);
        mSensorManager.registerListener(mEventListenerGyroscope,
                mSensorManager.getDefaultSensor(Sensor.TYPE_GYROSCOPE),
                SensorManager.SENSOR_DELAY_NORMAL);
    }
    @Override
    protected void onStop() {
        mSensorManager.unregisterListener(mEventListenerAccelerometer);
        mSensorManager.unregisterListener(mEventListenerGyroscope);
        super.onStop();
    }
}
```

四、 实验报告要求

写个程序检测手机拥有的所有传感器，列出手机中传感器的数量，每种传感器的设备名称、设备版本和生产厂商，并显示每种传感器随着时间的推移的读数。

实验报告中要包含以下几个部分：

1、实验目的
2、实验条件
3、实验原理
4、实验步骤分析
5、实验结果与总结
6、实验思考题

实验步骤要详细，关键步骤要有截图，运行结果也要有截图。