第 12 章　Android 特色开发，使用传感器

要说起 Android 的特色开发技术，除了基于位置的服务之外，传感器技术也绝对称得上是一点。现在每部 Android 手机里面都会内置有许多的传感器，它们能够监测到各种发生在手机上的物理事件，而我们只要灵活运用这些事件就可以编写出很多好玩的应用程序。那么话不多说，赶快开始我们本章的学习之旅吧。

12.1　传感器简介

手机中内置的传感器是一种微型的物理设备，它能够探测、感受到外界的信号，并按一定规律转换成我们所需要的信息。Android 手机通常都会支持多种类型的传感器，如光照传感右器、加速度传感器、地磁传感器、压力传感器、温度传感器等。

当然，Android 系统只是负责将这些传感器所输出的信息传递给我们，至于具体如何去利用这些信息就要充分发挥开发者的想象力了。目前市场上很多的程序都有使用到传感器的功能，比如最常见的赛车游戏，玩家可以通过旋转设备来控制赛车的前进方向，就像是在操作方向盘一样。除此之外，微信的摇一摇功能，手机指南针等软件也都是借助传感器来完成的。

不过，虽然 Android 系统支持十余种传感器的类型，但是手机里的传感器设备却是有限的，基本上不会有哪部手机能够支持全部的传感器功能。因此本章中我们只去学习最常见的几种传感器的用法，首先就从光照传感器开始吧。

12.2　光照传感器

光照传感器在 Android 中的应用还是比较常见的，比如系统就有个自动调整屏幕亮度的功能。它会检测手机周围环境的光照强度，然后对手机屏幕的亮度进行相应地调整，以此保证不管是在强光还是弱光下，手机屏幕都能够看得清。下面我们就来看下光照传感器到底是如何使用的吧。

12.2.1 光照传感器的用法

Android 中每个传感器的用法其实都比较类似，真的可以说是一通百通了。首先第一步要获取到 SensorManager 的实例，方法如下：

```
SensorManager senserManager = (SensorManager)
getSystemService(Context.SENSOR_SERVICE);
```

SensorManager 是系统所有传感器的管理器，有了它的实例之后就可以调用 getDefaultSensor()方法来得到任意的传感器类型了，如下所示：

```
Sensor sensor = senserManager.getDefaultSensor(Sensor.TYPE_LIGHT);
```

这里使用 Sensor.TYPE_LIGHT 常量来指定传感器类型，此时的 Sensor 实例就代表着一个光照传感器。Sensor 中还有很多其他传感器类型的常量，在后面几节中我们会慢慢学到。

接下来我们需要对传感器输出的信号进行监听，这就要借助 SensorEventListener 来实现了。SensorEventListener 是一个接口，其中定义了 onSensorChanged()和 onAccuracyChanged()这两个方法，如下所示：

```
SensorEventListener listener = new SensorEventListener() {

    @Override
    public void onAccuracyChanged(Sensor sensor, int accuracy) {
    }

    @Override
    public void onSensorChanged(SensorEvent event) {
    }

};
```

当传感器的精度发生变化时就会调用 onAccuracyChanged()方法，当传感器监测到的数值发生变化时就会调用 onSensorChanged()方法。可以看到 onSensorChanged()方法中传入了一个 SensorEvent 参数，这个参数里又包含了一个 values 数组，所有传感器输出的信息都是存放在这里的。

下面我们还需要调用 SensorManager 的 registerListener()方法来注册 SensorEventListener 才能使其生效，registerListener()方法接收三个参数，第一个参数就是 SensorEventListener 的实例，第二个参数是 Sensor 的实例，这两个参数我们在前面都已经成功得到了。第三个参数是用于表示传感器输出信息的更新速率，共有 SENSOR_DELAY_UI、SENSOR_DELAY_NORMAL、SENSOR_DELAY_GAME 和 SENSOR_DELAY_FASTEST 这四种值可选，它们的更新速率是依次递增的。因此，注册一个 SensorEventListener 就可以写成：

```
senserManager.registerListener(listener, senser, SensorManager.SENSOR_
DELAY_NORMAL);
```

另外始终要记得，当程序退出或传感器使用完毕时，一定要调用 unregisterListener ()方法将使用的资源释放掉，如下所示：

```
sensorManager.unregisterListener(listener);
```

好了，基本上用法就是这样，下面就让我们来动手尝试一下吧。

12.2.2　制作简易光照探测器

这里我们准备编写一个简易的光照探测器程序，使得手机可以检测到周围环境的光照强度。注意，由于模拟器中无法模拟出传感器的功能，因此仍然建议你将本章所写的代码都在手机上运行。

新建一个 LightSensorTest 项目，修改 activity_main.xml 中的代码，如下所示：

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <TextView
        android:id="@+id/light_level"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:textSize="20sp" />

</RelativeLayout>
```

布局文件同样是非常简单，只有一个 TextView 用于显示当前的光照强度，并让它在 RelativeLayout 中居中显示。

接着修改 MainActivity 中的代码，如下所示：

```
public class MainActivity extends Activity {

    private SensorManager sensorManager;

    private TextView lightLevel;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.activity_main);
        lightLevel = (TextView) findViewById(R.id.light_level);
        sensorManager = (SensorManager) getSystemService(Context.
SENSOR_SERVICE);
        Sensor sensor = sensorManager.getDefaultSensor(Sensor.TYPE_LIGHT);
        sensorManager.registerListener(listener, sensor, SensorManager.
SENSOR_DELAY_NORMAL);
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        if (sensorManager != null) {
            sensorManager.unregisterListener(listener);
        }
    }

    private SensorEventListener listener = new SensorEventListener() {

        @Override
        public void onSensorChanged(SensorEvent event) {
            // values数组中第一个下标的值就是当前的光照强度
            float value = event.values[0];
            lightLevel.setText("Current light level is " + value + " lx");
        }

        @Override
        public void onAccuracyChanged(Sensor sensor, int accuracy) {
        }

    };

}
```

这部分代码相信你理解起来一定轻而易举吧，因为这完全就是我们在上一小节中学到的用法。首先我们在 onCreate()方法中获取到了 TextView 控件的实例，然后按照上一小节中的步骤来注册光照传感器的监听器，这样当光照强度发生变化的时候就会调用 SensorEventListener 的 onSensorChanged()方法，而我们所需要的数据就是存放在 SensorEvent 的 values 数组中的。在这个例子当中，values 数组中只会有一个值，就是手机当前检测到的光照强度，以勒克斯为单位，我们直接将这个值显示到 TextView 上即可。最后不要忘记，在 onDestroy()方法中

还要调用 unregisterListener()方法来释放使用的资源。

现在运行一下程序，你将会在手机上看到当前环境下的光照强度，根据所处环境的不同，显示的数值有可能是几十到几百勒克斯。而如果你使用强光来照射手机的话，就有可能会达到上千勒克斯的光照强度，如图 12.1 所示。

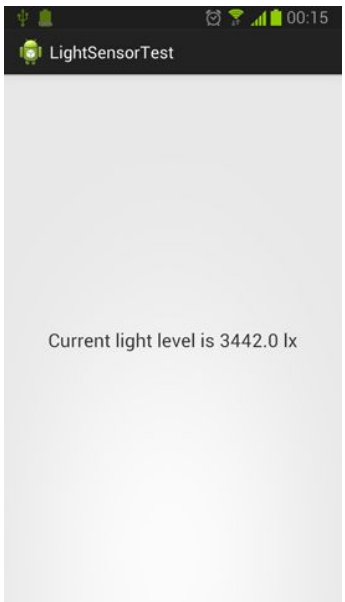

图　12.1

12.3　加速度传感器

相信你以前在上物理课的时候一定学习过加速度这个概念，它是一种用于描述物体运动速度改变快慢的物理量，以 m/s^2 为单位。而 Android 中的加速度传感器则是提供了一种机制，使得我们能够在应用程序中获取到手机当前的加速度信息，合理利用这些信息就可以开发出一些比较好玩的功能。

12.3.1　加速度传感器的用法

正如前面所说的一样，每种传感器的用法都是大同小异的，在上一节中你已经掌握了光照传感器的用法，因此，重复的部分我们就不再介绍了，这里在使用加速度传感器的时候只需要注意两点。第一，获取 Sensor 实例的时候要指定一个加速度传感器的常量，如下所示：

```
Sensor sensor = sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
```

第二，加速度传感器输出的信息同样也是存放在 SensorEvent 的 values 数组中的，只不过此时的 values 数组中会有三个值，分别代表手机在 X 轴、Y 轴和 Z 轴方向上的加速度信息。X 轴、Y 轴、Z 轴在空间坐标系上的含义如图 12.2 所示。

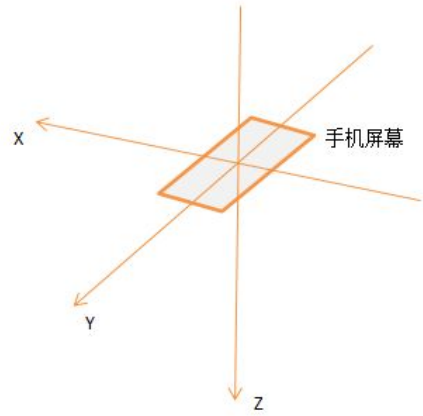

图　12.2

需要注意的是，由于地心引力的存在，你的手机无论在世界上任何角落都会有一个重力加速度，这个加速度的值大约是 9.8m/s^2。当手机平放的时候，这个加速度是作用在 Z 轴上的，当手机竖立起来的时候，这个加速度是作用在 Y 轴上的，当手机横立起来的时候，这个加速度是作用在 X 轴上的。

12.3.2　模仿微信摇一摇

接下来我们尝试利用加速度传感器来模仿一下微信的摇一摇功能。其实主体逻辑也非常简单，只需要检测手机在 X 轴、Y 轴和 Z 轴上的加速度，当达到了预定值的时候就可以认为用户摇动了手机，从而触发摇一摇的逻辑。那么现在问题在于，这个预定值应该设定为多少呢？由于重力加速度的存在，即使手机在静止的情况下，某一个轴上的加速度也有可能达到 9.8m/s^2，因此这个预定值必定是要大于 9.8m/s^2 的，这里我们就设定为 15m/s^2 吧。

新建一个 AccelerometerSensorTest 项目，然后修改 MainActivity 中的代码，如下所示：

```
public class MainActivity extends Activity {

    private SensorManager sensorManager;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.activity_main);
        sensorManager = (SensorManager) getSystemService
(Context.SENSOR_SERVICE);
        Sensor sensor = sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
        sensorManager.registerListener(listener, sensor, SensorManager.
SENSOR_DELAY_NORMAL);
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        if (sensorManager != null) {
            sensorManager.unregisterListener(listener);
        }
    }

    private SensorEventListener listener = new SensorEventListener() {

        @Override
        public void onSensorChanged(SensorEvent event) {
            // 加速度可能会是负值，所以要取它们的绝对值
            float xValue = Math.abs(event.values[0]);
            float yValue = Math.abs(event.values[1]);
            float zValue = Math.abs(event.values[2]);
            if (xValue > 15 || yValue > 15 || zValue > 15) {
                // 认为用户摇动了手机，触发摇一摇逻辑
                Toast.makeText(MainActivity.this, "摇一摇",
Toast.LENGTH_SHORT).show();
            }
        }

        @Override
        public void onAccuracyChanged(Sensor sensor, int accuracy) {
        }

    };

}
```

可以看到，这个例子还是非常简单的，我们在 onSensorChanged()方法中分别获取到了 X 轴、Y 轴和 Z 轴方向上的加速度值，并且由于加速度有可能是负值，所以这里又对获取到的

数据进行了绝对值处理。接下来进行了一个简单的判断，如果手机在 X 轴或 Y 轴或 Z 轴方向上的加速度值大于 15m/s^2，就认为用户摇动了手机，从而触发摇一摇的逻辑。当然，这里简单起见，我们只是弹出了一个 Toast 而已。

现在运行一下程序，并且摇动你的手机，就会看到有 Toast 提示出来了。

当然，这个程序只是一个非常非常简单的例子，你还可以在很多方面对它进行优化，比如说控制摇一摇的触发频率，使它短时间内不能连续触发两次等。更加缜密的逻辑就要靠你自己去完善了。

12.4 方向传感器

要说 Android 中另外一个比较常用的传感器应该就是方向传感器了。方向传感器的使用场景要比其他的传感器更为广泛，它能够准确地判断出手机在各个方向的旋转角度，利用这些角度就可以编写出像指南针、地平仪等有用的工具。另外，在本章开始时介绍的通过旋转设备来控制方向的赛车游戏，也是使用方向传感器来完成的。那么我们仍然还是先来看一下方向传感器的用法吧。

12.4.1 方向传感器的用法

通过前两节的学习，对于方向传感器的用法相信你已经成竹在胸了。没错，我们需要获取到一个用于表示方向传感器的 Sensor 实例，如下所示：

```
Sensor sensor = sensorManager.getDefaultSensor(Sensor.TYPE_ORIENTATION);
```

之后在 onSensorChanged()方法中通过 SensorEvent 的 values 数组，就可以得到传感器输出的所有值了。方向传感器会记录手机在所有方向上的旋转角度，如图 12.3 所示。

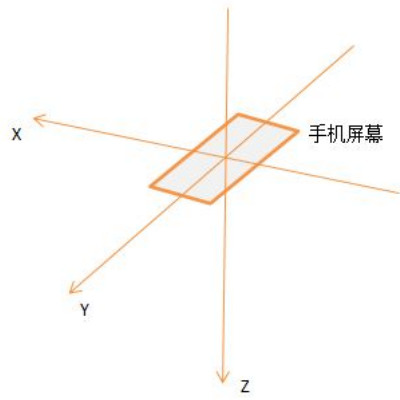

图 12.3

其中，values[0]记录着手机围绕 Z 轴的旋转角度，values[1] 记录着手机围绕 X 轴的旋转角度，values[2] 记录着手机围绕 Y 轴的旋转角度。

看起来很美好是吗？但遗憾的是，Android 早就废弃了 Sensor.TYPE_ORIENTATION 这种传感器类型，虽然代码还是有效的，但已经不再推荐这么写了。事实上，Android 获取手机旋转的方向和角度是通过加速度传感器和地磁传感器共同计算得出的，这也是 Android 目前推荐使用的方式。

首先我们需要分别获取到加速度传感器和地磁传感器的实例，并给它们注册监听器，如下所示：

```
    Sensor accelerometerSensor = sensorManager.getDefaultSensor(Sensor.
TYPE_ACCELEROMETER);
    Sensor magneticSensor = sensorManager.getDefaultSensor(Sensor.
TYPE_MAGNETIC_FIELD);
    sensorManager.registerListener(listener, accelerometerSensor,
SensorManager.SENSOR_DELAY_GAME);
    sensorManager.registerListener(listener, magneticSensor,
SensorManager.SENSOR_DELAY_GAME);
```

由于方向传感器的精确度要求通常都比较高，这里我们把传感器输出信息的更新速率提高了一些，使用的是 SENSOR_DELAY_GAME。

接下来在 onSensorChanged()方法中可以获取到 SensorEvent 的 values 数组，分别记录着加速度传感器和地磁传感器输出的值。然后将这两个值传入到 SensorManager 的 getRotationMatrix()方法中就可以得到一个包含旋转矩阵的 R 数组，如下所示：

```
    SensorManager.getRotationMatrix(R, null, accelerometerValues, magneticValues);
```

其中第一个参数 R 是一个长度为 9 的 float 数组，getRotationMatrix()方法计算出的旋转数据就会赋值到这个数组当中。第二个参数是一个用于将地磁向量转换成重力坐标的旋转矩阵，通常指定为 null 即可。第三和第四个参数则分别就是加速度传感器和地磁传感器输出的 values 值。

得到了 R 数组之后，接着就可以调用 SensorManager 的 getOrientation()方法来计算手机的旋转数据了，如下所示：

```
    SensorManager.getOrientation(R, values);
```

values 是一个长度为 3 的 float 数组，手机在各个方向上的旋转数据都会被存放到这个数组当中。其中 values[0]记录着手机围绕着图 12.3 中 Z 轴的旋转弧度，values[1]记录着手机围绕 X 轴的旋转弧度，values[2]记录着手机围绕 Y 轴的旋转弧度。

注意这里计算出的数据都是以弧度为单位的，因此如果你想将它们转换成角度还需要调用如下方法：

```
Math.toDegrees(values[0]);
```

好了，基本的用法就是如此，下面我们来实际操作一下吧。

12.4.2 制作简易指南针

有了上一小节的知识储备，相信完成一个简易的指南针程序对于你来说已经不是一件难事了。其实指南针的实现原理并不复杂，我们只需要检测到手机围绕 Z 轴的旋转角度，然后对这个值进行处理就可以了。

新建一个 CompassTest 项目，修改 MainActivity 中的代码，如下所示：

```
public class MainActivity extends Activity {

    private SensorManager sensorManager;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        sensorManager = (SensorManager) getSystemService(Context.
SENSOR_SERVICE);
        Sensor magneticSensor = sensorManager.getDefaultSensor(Sensor.
TYPE_MAGNETIC_FIELD);
        Sensor accelerometerSensor = sensorManager.getDefaultSensor(
Sensor.TYPE_ACCELEROMETER);
        sensorManager.registerListener(listener, magneticSensor,
SensorManager.SENSOR_DELAY_GAME);
        sensorManager.registerListener(listener, accelerometerSensor,

    SensorManager.SENSOR_DELAY_GAME);
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        if (sensorManager != null) {
            sensorManager.unregisterListener(listener);
        }
    }

    private SensorEventListener listener = new SensorEventListener() {
```

```
        float[] accelerometerValues = new float[3];

        float[] magneticValues = new float[3];

        @Override
        public void onSensorChanged(SensorEvent event) {
            // 判断当前是加速度传感器还是地磁传感器
            if (event.sensor.getType() == Sensor.TYPE_ACCELEROMETER) {
                // 注意赋值时要调用clone()方法
                accelerometerValues = event.values.clone();
            } else if (event.sensor.getType() == Sensor.TYPE_MAGNETIC_FIELD) {
                // 注意赋值时要调用clone()方法
                magneticValues = event.values.clone();
            }
            float[] R = new float[9];
            float[] values = new float[3];
            SensorManager.getRotationMatrix(R, null, accelerometerValues,
magneticValues);
            SensorManager.getOrientation(R, values);
            Log.d("MainActivity", "value[0] is " + Math.toDegrees(values[0]));
        }

        @Override
        public void onAccuracyChanged(Sensor sensor, int accuracy) {
        }

    };

}
```

代码不算很长，相信你理解起来应该毫不费力。在 onCreate()方法中我们分别获取到了加速度传感器和地磁传感器的实例，并给它们注册了监听器。然后在 onSensorChanged()方法中进行判断，如果当前 SensorEvent 中包含的是加速度传感器，就将 values 数组赋值给 accelerometerValues 数组，如果当前 SensorEvent 中包含的是地磁传感器，就将 values 数组赋值给 magneticValues 数组。注意在赋值的时候一定要调用一下 values 数组的 clone()方法，不然 accelerometerValues 和 magneticValues 将会指向同一个引用。

接下来我们分别创建了一个长度为 9 的 R 数组和一个长度为 3 的 values 数组，然后调用 getRotationMatrix()方法为 R 数组赋值，再调用 getOrientation()方法为 values 数组赋值，这时

values 中就已经包含手机在所有方向上旋转的弧度了。其中 values[0]表示手机围绕 Z 轴旋转的弧度，这里我们调用 Math.toDegrees()方法将它转换成角度，并打印出来。

现在运行一下程序，并围绕 Z 轴旋转手机，旋转的角度就会源源不断地在 LogCat 中打印出来了，如图 12.4 所示。

Tag	Text
MainActivity	value[0] is 169.21651600771855
MainActivity	value[0] is 169.4880979894966
MainActivity	value[0] is 169.74341046067167
MainActivity	value[0] is 170.01637214066204
MainActivity	value[0] is 170.27384295161485
MainActivity	value[0] is 170.30571261428213

图 12.4

values[0]的取值范围是–180 度到 180 度，其中±180 度表示正南方向，0 度表示正北方向，–90 度表示正西方向，90 度表示正东方向。

虽然目前我们已经得到了这些数值，但是想要通过它们来判断手机当前的方向显然是一件伤脑筋的事情，因此我们还要想办法将当前的方向直观地显示出来。毫无疑问，最直观的方式当然是通过罗盘和指针来进行显示了，那么下面我们就来继续完善 CompassTest 这个项目。

这里我事先准备好了两张图片 compass.png 和 arrow.png，分别用于作为指南针的背景和指针，如图 12.5 所示：

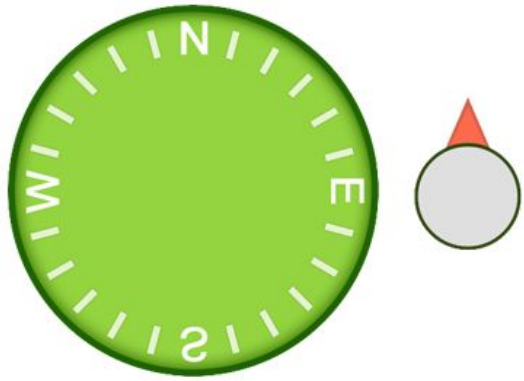

图 12.5

然后修改 activity_main.xml 中的代码，如下所示：

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <ImageView
        android:id="@+id/compass_img"
        android:layout_width="250dp"
        android:layout_height="250dp"
        android:layout_centerInParent="true"
        android:src="@drawable/compass" />

    <ImageView
        android:id="@+id/arrow_img"
        android:layout_width="60dp"
        android:layout_height="110dp"
        android:layout_centerInParent="true"
        android:src="@drawable/arrow" />

</RelativeLayout>
```

可以看到，我们在 RelativeLayout 中放入了两个 ImageView 控件，并且让这两个控件都居中显示，ImageView 中显示的图片分别就是指南针的背景和指针。

接下来修改 MainActivity 中的代码，如下所示：

```
public class MainActivity extends Activity {

    private SensorManager sensorManager;

    private ImageView compassImg;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        compassImg = (ImageView) findViewById(R.id.compass_img);
        ……
    }
    ……
    private SensorEventListener listener = new SensorEventListener() {
```

```
        float[] accelerometerValues = new float[3];

        float[] magneticValues = new float[3];

        private float lastRotateDegree;

        @Override
        public void onSensorChanged(SensorEvent event) {
            // 判断当前是加速度传感器还是地磁传感器
            if (event.sensor.getType() == Sensor.TYPE_ACCELEROMETER) {
                // 注意赋值时要调用clone()方法
                accelerometerValues = event.values.clone();
            } else if (event.sensor.getType() == Sensor.TYPE_MAGNETIC_FIELD) {
                // 注意赋值时要调用clone()方法
                magneticValues = event.values.clone();
            }
            float[] values = new float[3];
            float[] R = new float[9];
            SensorManager.getRotationMatrix(R, null, accelerometerValues,
magneticValues);
            SensorManager.getOrientation(R, values);
            // 将计算出的旋转角度取反，用于旋转指南针背景图
            float rotateDegree = -(float) Math.toDegrees(values[0]);
            if (Math.abs(rotateDegree - lastRotateDegree) > 1) {
                RotateAnimation animation = new RotateAnimation
(lastRotateDegree, rotateDegree, Animation.RELATIVE_TO_SELF, 0.5f, Animation.
RELATIVE_TO_SELF, 0.5f);
                animation.setFillAfter(true);
                compassImg.startAnimation(animation);
                lastRotateDegree = rotateDegree;
            }
        }
        ……
    };

}
```

这里首先我们在 onCreate()方法中获取到了 ImageView 的实例，它是用于显示指南针的背景图的。然后在 onSensorChanged()方法中使用到了旋转动画技术，我们创建了一个 RotateAnimation 的实例，并给它的构造方法传入了六个参数，第一个参数表示旋转的起始角

度，第二个参数表示旋转的终止角度，后面四个参数用于指定旋转的中心点。这里我们把从传感器中获取到的旋转角度取反，传递给 RotateAnimation，并指定旋转的中心点为指南针背景图的中心，然后调用 ImageView 的 startAnimation ()方法来执行旋转动画。

好了，代码就是这么多，现在我们重新运行一下程序，然后随意旋转手机，指南针的背景图也会跟着一起转动，如图 12.6 所示。

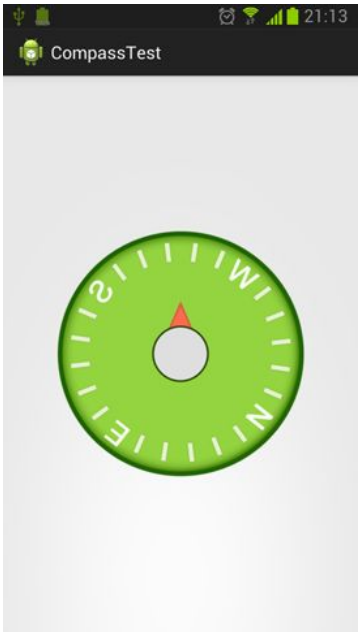

图　12.6

这样，一个简易的指南针程序也就完成了，下面我们来回顾一下本章所学的内容吧。

12.5　小结与点评

在本章的 Android 特色开发中，我们学会了多种传感器的使用方式，只要巧妙地利用这些传感器就可以编写出许多有趣的应用程序和游戏。而且，千万不要让你的思想受限于本章所编写的几个例子当中，你完全可以充分地发挥你的想像力，配合强大的传感器功能，制作出更加创意十足的应用程序。

当然了，Android 中支持的传感器远远不只这些，还有压力传感器、温度传感器、陀螺仪传感器等，不过由于这些传感器都不太常用，而且不少 Android 手机中都没有嵌入这些传感器，因此在本章中我们就没有进行讲解。但是正如前面所说，每种传感器的用法都比较相似，即使让你去自学其他几种传感器的用法，相信也已经是轻而易举了吧。

现在你已经足足学习了十二章的内容，对 Android 应用程序开发的理解应该是比较深刻了。目前系统性的知识几乎都已经讲完了，但是还有一些零散的高级技巧在等待着你，那么就让我们赶快进入到下一章的学习当中。

经验值：+100000　　升级！（由头领鸟升级至鹰）　　目前经验值：364905

级别：鹰

赢得宝物：战胜神界传感器地鼠。拾取传感器地鼠掉落的宝物，大量游戏币。其实我战胜传感器地鼠完全是个误会，因为我当时只是在一个小镇的游乐场里开心地玩打地鼠游戏，正当我玩得起劲时，这帮被打得嘣嘣直响的小地鼠们突然从游戏机里蹦了出来，说恭喜我我战胜了他们，并主动赠送了我好些游戏币，惊讶之余弄得我好不尴尬。他们说他们生存的意义就是被橡皮锤打脑袋，而现在网络这么发达，新鲜玩意儿太多，已经许久没有人来打他们了，他们感到已经失去了生活的目标，多亏我这个异乡人今天帮他们重新找回了自信。我说此地不留爷，自有留爷处，或许你们可以在人界闯出一片天地，那里小孩子特别多，一定会非常喜欢敲打你们的。这群小家伙茅塞顿开，激动不已，又给了我更多的游戏币。我向他们道别，将游戏币兑换成盘缠。正准备继续前进时，觉得背部有点痒，难道是与这帮地鼠走得太近，不幸招惹上了跳蚤。算了，不管它。继续前进。且慢！少侠，请留步。谁？谁在对我说话？一歪头，旁边树上正蹲着那只鬼鬼祟祟的松鼠。

“难道你没意识到自己的变化么？”，如果我没记错，这应该是这位跟踪了我一路的家伙第一次对我说话。

“除了招了跳蚤，别的没啥，咋了？”

“不是跳蚤，是别的东西。”

别的东西？难道是一路风餐露宿的感染了真菌，得了皮肤病？让这家伙一说，我不由得紧张起来。我伸手往背后瘙痒处摸索，摸到的东西吓了我一大跳，是一对翅膀！而且还不是什么正经翅膀，是那种比肯德基奥尔良烤翅大不了多少的肉肉的小翅膀！

“我怎么了？！我到底得了什么病？！难道是得了神界禽流感？？？”

“不！别担心，这意味着你进入了更高阶的层次。你现在是鹰了！现在这翅膀还太小，派不上什么用场，但随着你的修行，翅膀会发挥它应有的作用，并且……，算了，不说了。到时你自然会知道。”

“是什么作用啊，是能飞嘛？还是像鸵鸟那样只是装饰？”我刚问完，他却又打了个响指消失了。

我又摸了摸小翅膀，很小，很软和。顶着一对小鸡翅，我继续前进。顺便说一下，我现在已经可以用意念来关闭和开启我自身的光芒了。