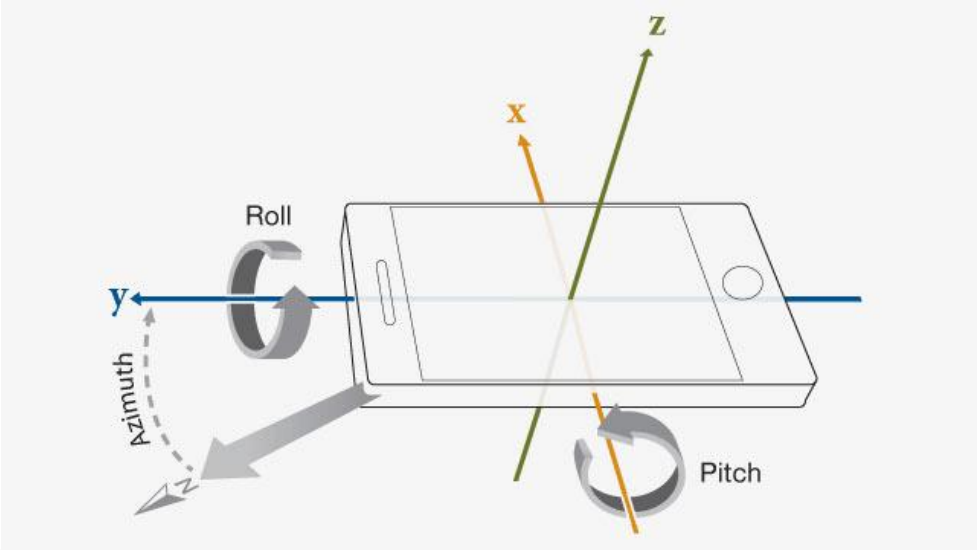
Android Sensors

Overview

- Most Android-powered devices have built-in sensors that measure motion, orientation, and various environmental conditions
- These sensors are capable of providing raw data with high precision and accuracy
- The Android platform supports three broad categories of sensors

Motion sensors

- These sensors measure acceleration forces and rotational forces along three axes. This category includes accelerometers, gravity sensors, gyroscopes, and rotational vector sensors

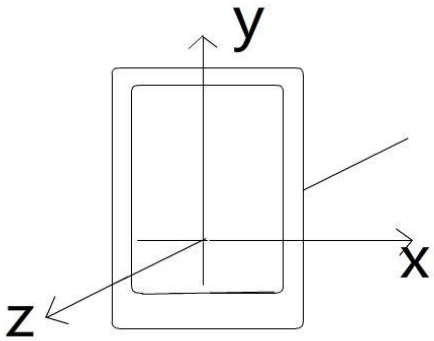


Position Sensors 1/2

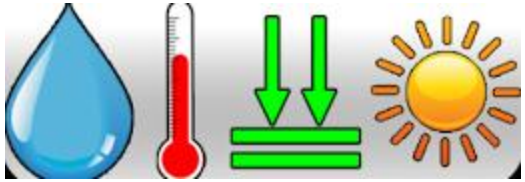
- The Android platform provides two sensors that let you determine the position of a device: the geomagnetic field sensor and the orientation sensor
- The Android platform also provides a sensor that lets you determine how close the face of a device is to an object, known as the proximity sensor

Position Sensors 2/2

- The geomagnetic field sensor and the proximity sensor are hardware-based
- Handset manufacturers usually include a proximity sensor to determine when a handset is being held close to a user's face
- The orientation sensor is software-based and derives its data from the accelerometer and the geomagnetic field sensor



Environment Sensors



- The Android platform provides four sensors that let you monitor various environmental properties
- You can use these sensors to monitor relative ambient humidity, illuminance, ambient pressure, and ambient temperature near an Android-powered device
- All four environment sensors are hardware-based and are available only if a device manufacturer has built them into a device

Supported Environment Sensors

Sensor	Sensor event data	Units of measure	Data description
`TYPE_AMBIENT_TEMPERATURE`	`event.values[0]`	°C	Ambient air temperature.
`TYPE_LIGHT`	`event.values[0]`	lx	Illuminance.
`TYPE_PRESSURE`	`event.values[0]`	hPa or mbar	Ambient air pressure.
`TYPE_RELATIVE_HUMIDITY`	`event.values[0]`	%	Ambient relative humidity.
`TYPE_TEMPERATURE`	`event.values[0]`	°C	Device temperature.[1]

Sensor Framework

- Determine which sensors are available on a device
- Determine an individual sensor's capabilities, such as its maximum range, manufacturer, power requirements, and resolution
- Acquire raw sensor data and define the minimum rate at which you acquire sensor data
- Register and unregister sensor event listeners that monitor sensor changes

Sensor availability by platform

Sensor	Android 4.0 (API Level 14)	Android 2.3 (API Level 9)	Android 2.2 (API Level 8)	Android 1.5 (API Level 3)
TYPE_ACCELEROMETER	Yes	Yes	Yes	Yes
TYPE_AMBIENT_TEMPERATURE	Yes	n/a	n/a	n/a
TYPE_GRAVITY	Yes	Yes	n/a	n/a
TYPE_GYROSCOPE	Yes	Yes	n/a[1]	n/a[1]
TYPE_LIGHT	Yes	Yes	Yes	Yes
TYPE_LINEAR_ACCELERATION	Yes	Yes	n/a	n/a
TYPE_MAGNETIC_FIELD	Yes	Yes	Yes	Yes
TYPE_ORIENTATION	Yes[2]	Yes[2]	Yes[2]	Yes
TYPE_PRESSURE	Yes	Yes	n/a[1]	n/a[1]
TYPE_PROXIMITY	Yes	Yes	Yes	Yes
TYPE_RELATIVE_HUMIDITY	Yes	n/a	n/a	n/a
TYPE_ROTATION_VECTOR	Yes	Yes	n/a	n/a
TYPE_TEMPERATURE	Yes[2]	Yes	Yes	Yes

SensorManager

- SensorManager lets you access the device's sensors. Get an instance of this class by calling Context.getSystemService() with the argument SENSOR_SERVICE
- Always make sure to disable sensors you don't need, especially when your activity is paused
- Note that the system will not disable sensors automatically when the screen turns off

Creating an instance of the SensorManager class

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

Creating an instance of the Sensor Class

```
private SensorManager mSensorManager;
private Sensor mSensor;
  ...
mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
mSensor = mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
```

List every sensor on a device

```
List<Sensor> deviceSensors = mSensorManager.getSensorList(Sensor.TYPE_ALL);
```

- If you want to list all of the sensors of a given type, you could use another constant instead of TYPE_ALL such as TYPE_GYROSCOPE, TYPE_LINEAR_ACCELERATION, or TYPE_GRAVITY.

SensorEventListener

- public interface
- Used for receiving notifications from the SensorManager when sensor values have changed

Public Methods	
abstract void	onAccuracyChanged (Sensor sensor, int accuracy) Called when the accuracy of the registered sensor has changed.
abstract void	onSensorChanged (SensorEvent event) Called when sensor values have changed.

Register a listener

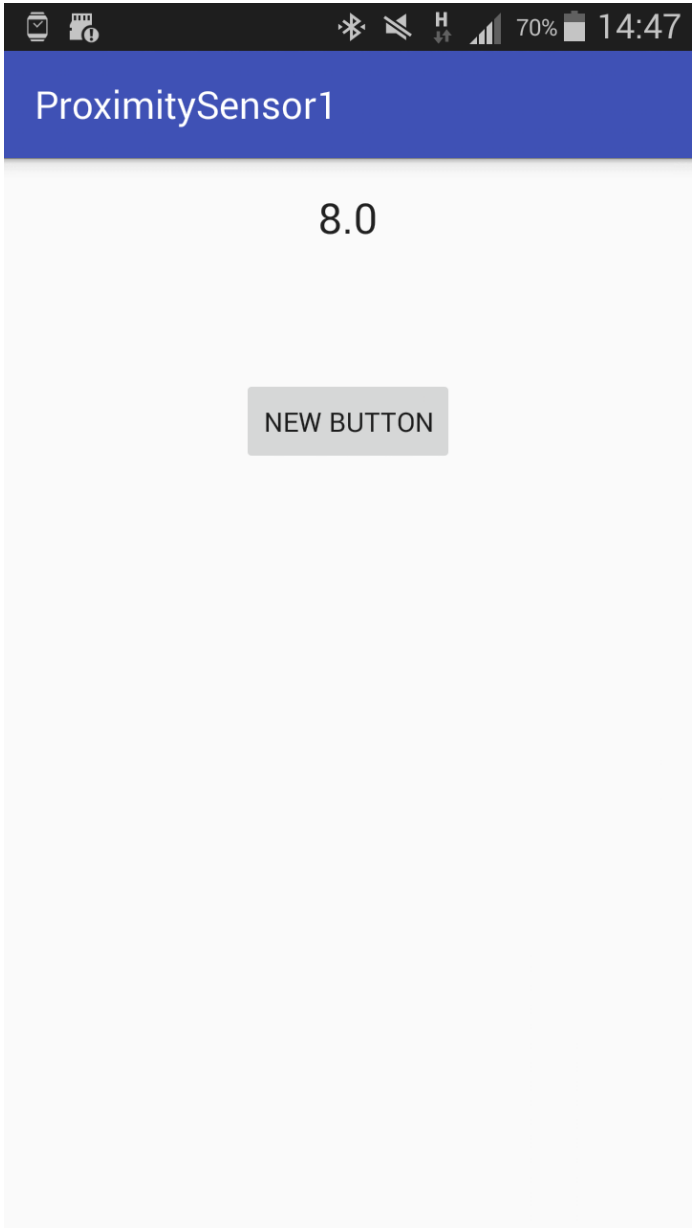
```
public boolean registerListener (SensorEventListener listener, Sensor sensor, int samplingPeriodUs)
```

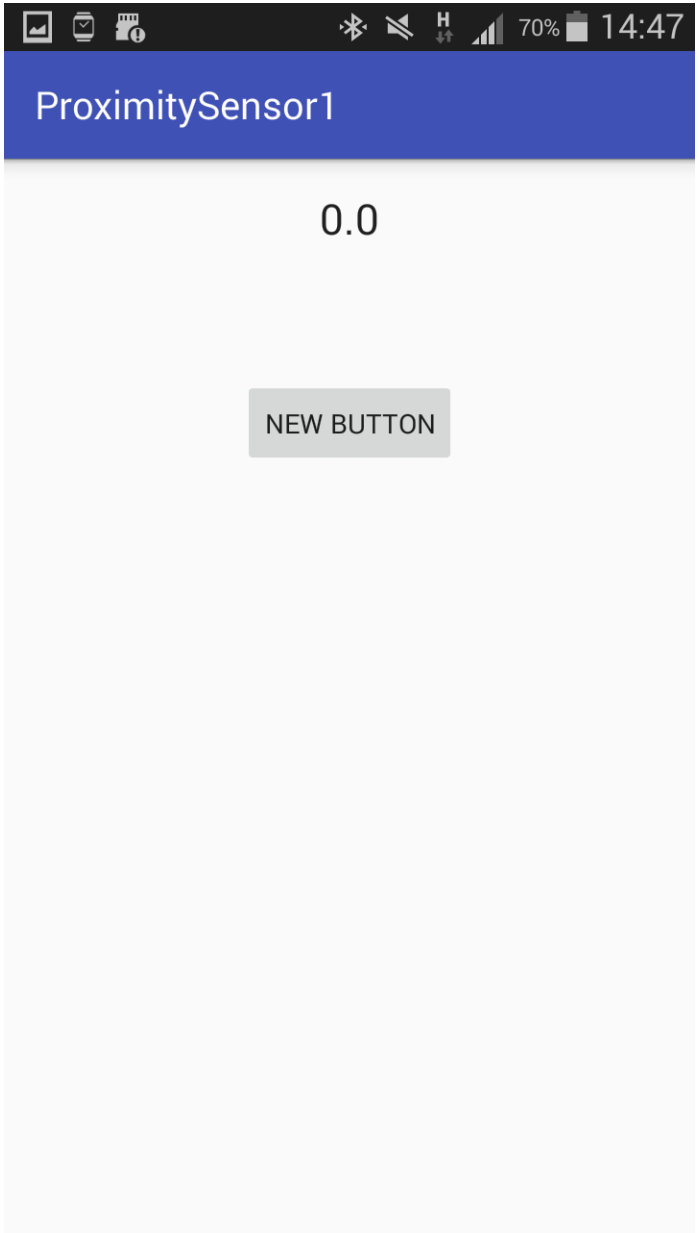
- Registers a SensorEventListener for the given sensor at the given sampling frequency
- The events will be delivered to the provided SensorEventListener as soon as they are available

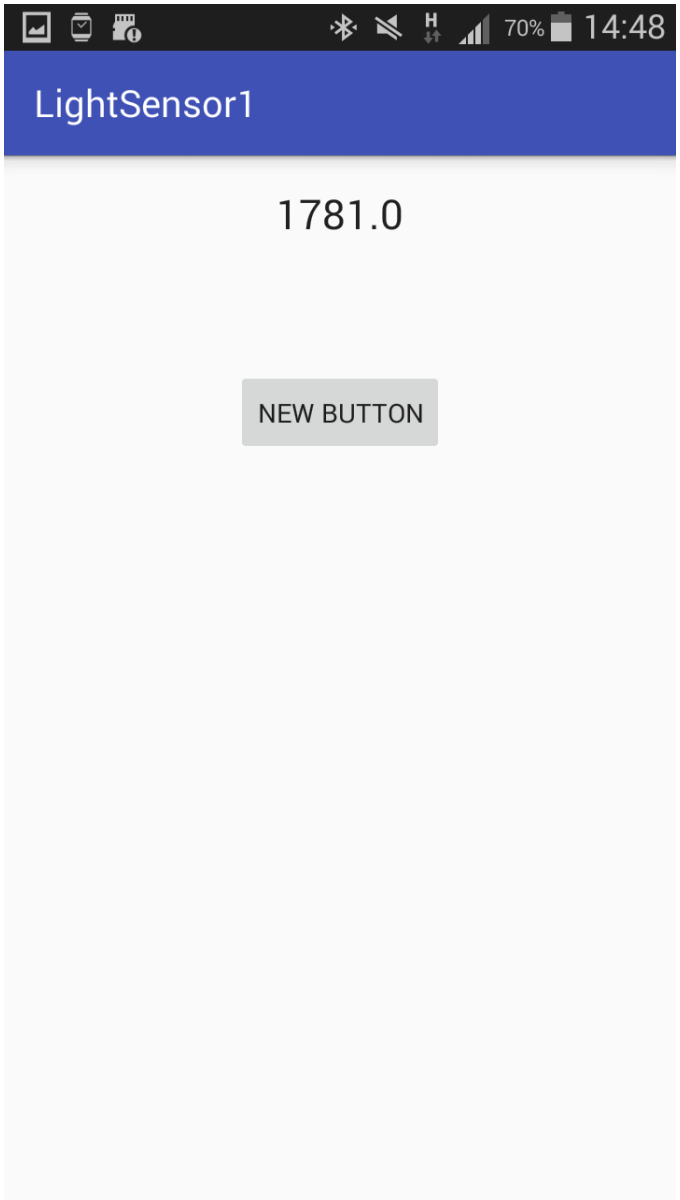
SensorEvent

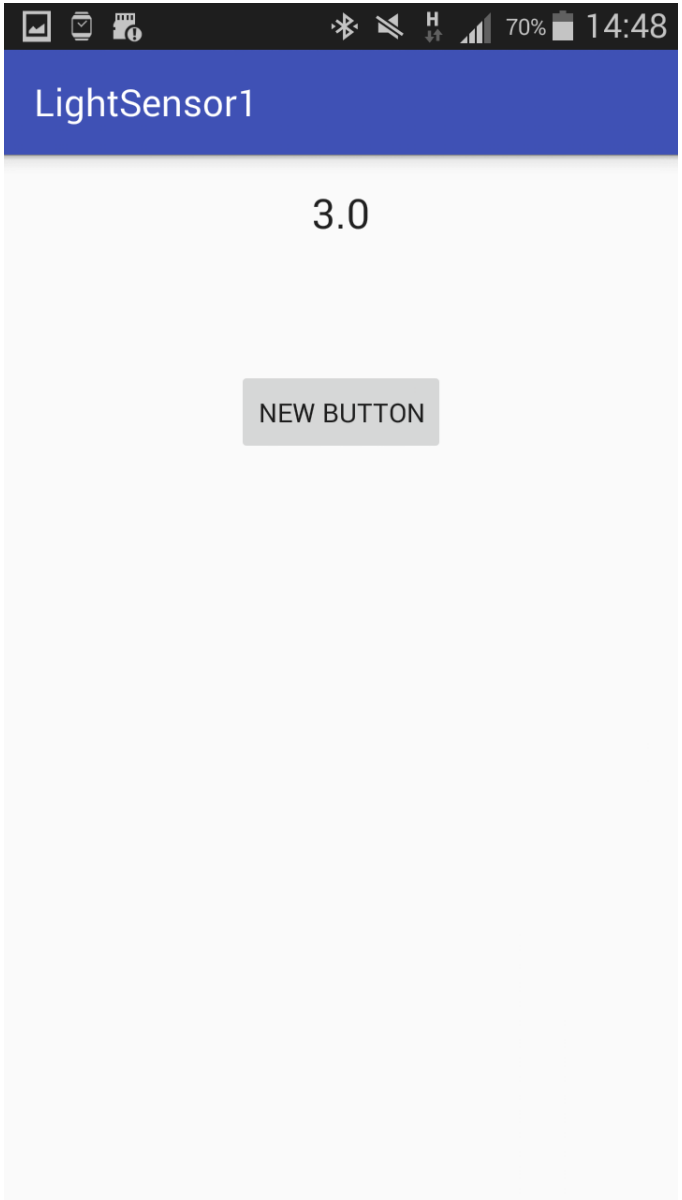
- public class
- This class represents a Sensor event and holds information such as the sensor's type, the time-stamp, accuracy and of course the sensor's data

Fields		
public int	accuracy	The accuracy of this event.
public Sensor	sensor	The sensor that generated this event.
public long	timestamp	The time in nanosecond at which the event happened
public final float[]	values	The length and contents of the `values` array depends on which `sensor` type is being monitored (see also `SensorEvent` for a definition of the coordinate system used).


70%
14:47
ProximitySensor1
8.0
NEW BUTTON


70%
14:47
ProximitySensor1
0.0
NEW BUTTON


70%
14:48
LightSensor1
1781.0
NEW BUTTON


70%
14:48
LightSensor1
3.0
NEW BUTTON