Using the Accelerometer and Compass

Input based on movement and orientation is an exciting innovation for mobile applications. It's a technique that has become possible thanks to the incorporation of compass and accelerometer sensors in modern devices.

Accelerometers and compasses are used to provide functionality based on changes in device orientation and movement. A recent trend is to use this functionality to provide alternative input techniques from more traditional touch-screen-, trackball-, and keyboard-based approaches. In recent years, these sensors have become increasingly common, having found their way into game controllers like the Nintendo Wii and mobile handsets like the Apple iPhone.

The availability of compass and accelerometer values depends on the hardware upon which your application runs. When available, they are exposed through the SensorManager class, allowing you to:

- ❑ Determine the current orientation of the hardware.
- ❑ Monitor for changes in orientation.
- ❑ Know which direction the user is facing.
- ❑ Monitor acceleration — changes in movement speed — in any direction: vertically, laterally, or longitudinally.

This opens some intriguing possibilities for your applications. By monitoring orientation, direction, and movement, you can:

- ❑ Use the compass and accelerometer to determine your speed and direction. Used with the maps and location-based services, you can create interfaces that incorporate direction and movement as well as location.
- ❑ Create User Interfaces that adjust dynamically to suit the orientation of your device. Android already alters the native screen orientation when the device is rotated from portrait to landscape or vice versa.
- ❑ Monitor for rapid acceleration to detect if a device has been dropped or thrown.
- ❑ Measure movement or vibration. For example, you could create an application that lets you lock your device; if any movement is detected while it's locked, it could send an alert IM message that includes its current location.
- ❑ Create User Interface controls that use physical gestures and movement as input.

Introducing Accelerometers

Accelerometers, as their name suggests, are used to measure acceleration. *Acceleration* is defined as the rate of change of velocity, so they measure how quickly the speed of the device is changing in a given direction. Using an accelerometer, you can detect movement and, more usefully, the rate of change of the speed of that movement.

It's important to note that accelerometers do not *measure velocity, so you can't measure speed directly based on a single accelerometer reading. Instead, you need to measure changes in acceleration over time.*

Generally, you'll be interested in acceleration changes relative to a rest state, or rapid movement (signify ed by rapid changes in acceleration) such as gestures used for user input. In the former case, you'll often need to calibrate the device to calculate the initial orientation and acceleration to take those effects into account in future results.

Accelerometers are unable to differentiate between acceleration due to movement and gravity. As a result, an accelerometer detecting acceleration on the Z-axis (up/down) will read –9.8 m/s2 when it's at rest (this value is available as the SensorManager.STANDARD_GRAVITY *constant).*

Detecting Acceleration Changes

Acceleration can be measured along three directional axes: forward–backward (longitudinal), left–right (lateral), and up–down (vertical). The Sensor Manager reports sensor changes in all three directions (as illustrated in Figure 10-1):

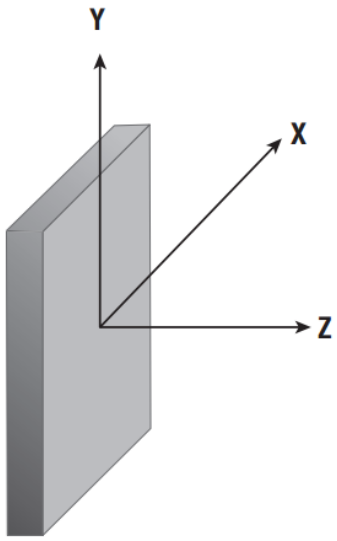

Figure 10-1

❑ **Vertical** Upward or downward, where positive represents upward movement such as the device being lifted up.

❑ **Longitudinal** Forward or backward acceleration, where forward acceleration is positive. This represents a device fl at on its back, facing up, and in portrait orientation being moved along the desk in the direction of the top of the device.

❑ **Lateral** Sideways (left or right) acceleration, where positive values represent movement toward the right of the device, and negative values show movement toward the left. In the same confi guration as described in longitudinal movement, positive lateral movement would be created by moving the device along the surface to your right.

The Sensor Manager considers the device "at rest" when it is sitting face up on a fl at surface in portrait orientation.

As described previously, you can monitor changes in acceleration using Sensor Listeners. Register an extension of the SensorListener class with the Sensor Manager, using the SENSOR_ACCELEROMETER flag to request updates of accelerometer values and a sensor update rate as shown in the following code snippet:

```
SensorManager sm = (SensorManager)getSystemService(Context.SENSOR_SERVICE);
sm.registerListener(mySensorListener,
SensorManager.SENSOR_ACCELEROMETER, SensorManager.SENSOR_DELAY_UI);
```

Your Sensor Listener must implement the onSensorChanged method that will be triggered when the changes in acceleration along any of the three axes described previously are detected.

The onSensorChanged method receives a float array that contains the current acceleration along all three axes in smoothed and raw formats. The Sensor Manager includes index constants that you can use to extract the acceleration value you require, as shown in the following code snippet:

```
SensorListener mySensorListener = new SensorListener() {
public void onSensorChanged(int sensor, float[] values) {
if (sensor == SensorManager.SENSOR_ACCELEROMETER) {
float xAxis = values[SensorManager.DATA_X];
float yAxis = values[SensorManager.DATA_Y];
float zAxis = values[SensorManager.DATA_Z];
float raw_xAxis = values[SensorManager.RAW_DATA_X];
float raw_yAxis = values[SensorManager.RAW_DATA_Y];
float raw_zAxis = values[SensorManager.RAW_DATA_Z];
// TODO apply the acceleration changes to your application.
}}
public void onAccuracyChanged(int sensor, int accuracy) { }
};
```