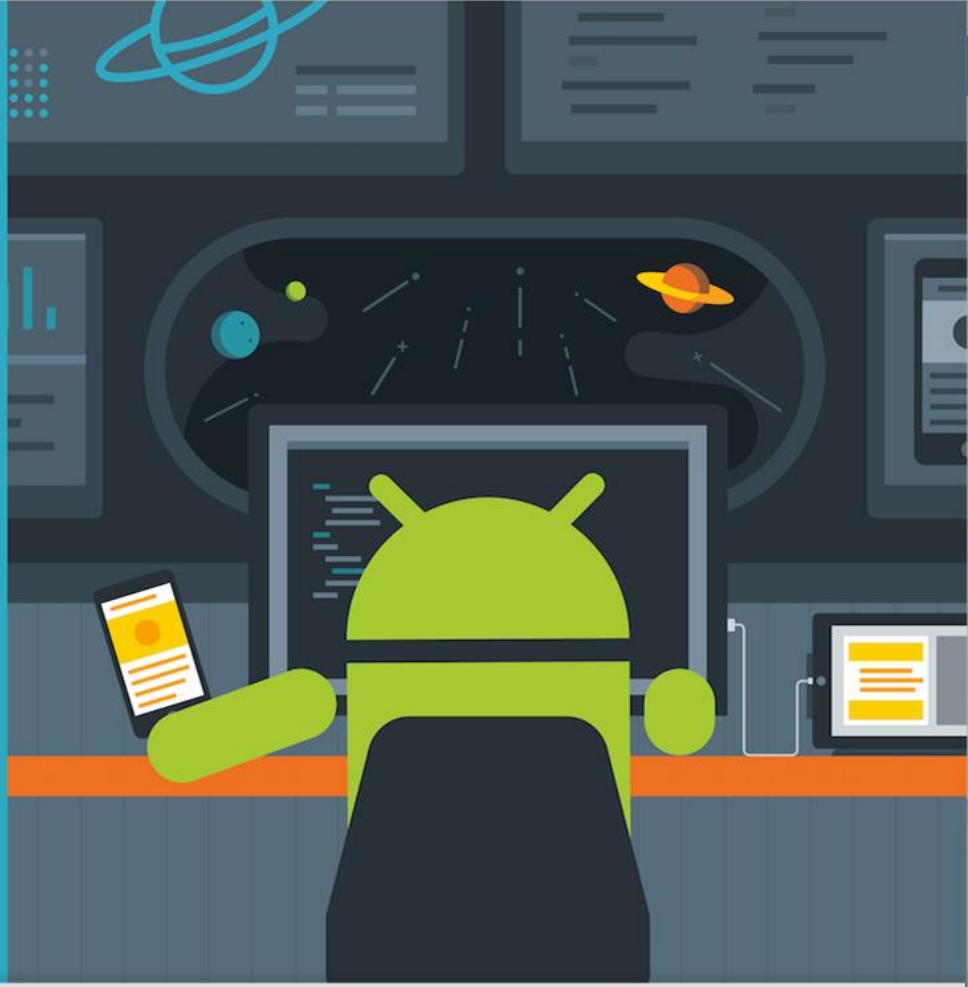
Advanced Android Development

Sensors

Lesson 3



3.2 Motion and position sensors

Monitor device movement or position in space

Contents

- Overview of motion and position sensors
- Determining device orientation
- Understanding device rotation
- Using motion sensors
- Using position sensors

Overview

Motion and position sensors

Motion and position sensors

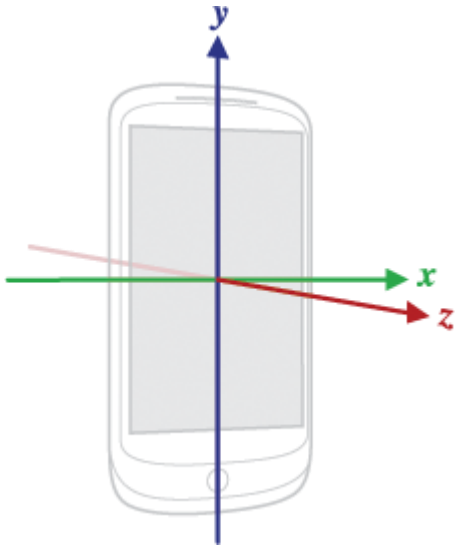
- Motion and position sensors monitor device movement or position in space respectively
- Both return multi-dimensional arrays of sensor values for each `SensorEvent`
  - Example: Accelerometer returns acceleration force data for 3 coordinate axes (*x*, *y*, *z*) relative to device

Coordinate systems

- **Device coordinate system** : Some sensors use device coordinate system relative to the device
  - Example: Accelerometers
- **Earth coordinate system** : Other sensors use Earth coordinate system relative to Earth surface
  - Example: Magnetometer

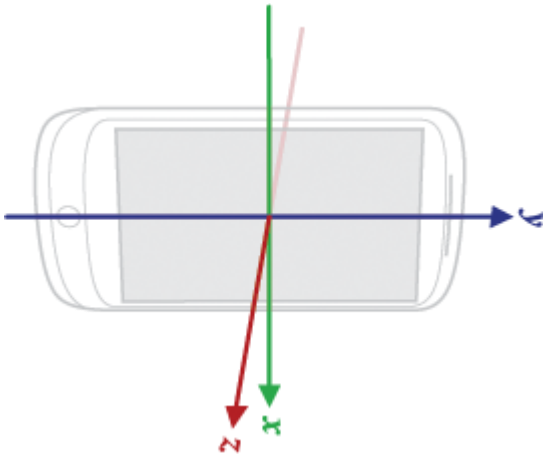
Device coordinates (1)

- Relative to physical device regardless of device position in the world
- *x* is horizontal and points right
- *y* is vertical and points up
- *z* points toward outside of screen
- Negative *z* points behind screen



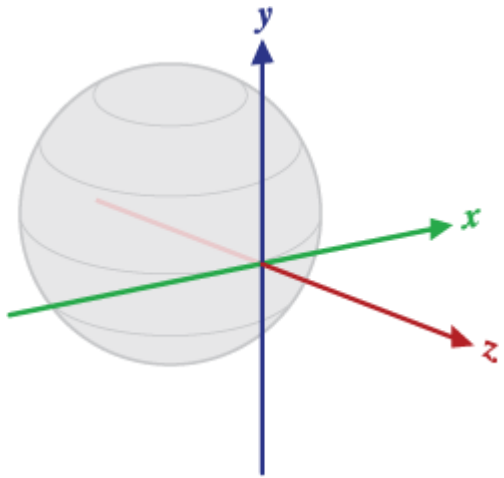
Device coordinates (2)

- Relative to the device screen when device is in its default orientation
- Axes are not swapped when orientation changes by rotation
- App must transform incoming sensor data to match rotation



Earth coordinates

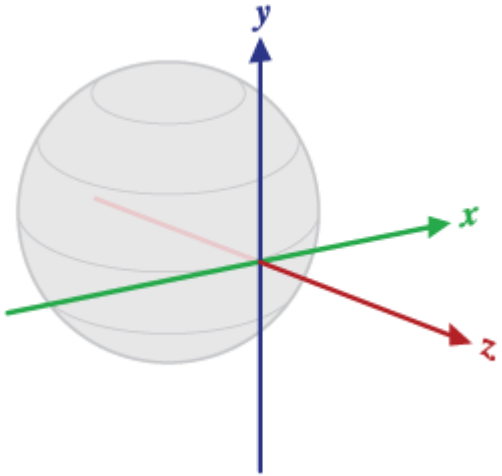
- *y* points to magnetic north along Earth's surface
- *x* is 90 degrees from *y*, pointing east
- *z* extends up into space
- Negative *z* extends down into ground



Determining device orientation

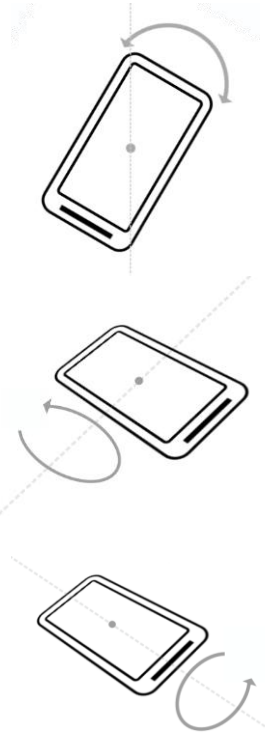
Device orientation

- Position of device relative to Earth's coordinates (*y* points to magnetic north)
- Determine by using accelerometer and geomagnetic field sensor with methods in `SensorManager`



Components of orientation

- Azimuth
  - Angle between device's compass direction and magnetic north
- Pitch
  - Angle between plane parallel to device's screen and plane parallel to ground
- Roll
  - Angle between plane perpendicular to device's screen and plane perpendicular to ground



SensorManager methods

- `getRotationMatrix()` generates rotation matrix from accelerometer and geomagnetic field sensor
  - Translates sensor data from device coordinates to Earth coordinates
- `getOrientation()` uses rotation matrix to compute angles of device's orientation

Example: Determine orientation

```
private SensorManager mSensorManager;
...
// Rotation matrix based on current readings.
final float[] rotationMatrix = new float[9];
mSensorManager.getRotationMatrix(rotationMatrix, null,
                 accelerometerReading, magnetometerReading);

// Express updated rotation matrix as 3 orientation angles.
final float[] orientationAngles = new float[3];
mSensorManager.getOrientation(rotationMatrix,
                                        orientationAngles);
```

Understanding device rotation

Transform coordinates for rotation

If app draws views based on sensor data:

- Screen or activity coordinate system rotates with device
- Sensor coordinate system doesn't rotate
- Need to transform sensor coordinates to activity coordinates

Handle device and activity rotation

1. Query device orientation with `getRotationMatrix()`
2. Remap rotation matrix from sensor data to activity coordinates with `remapCoordinateSystem()`

Returned from getRotation()

Integer constants:

- `ROTATION_0`: Default (portrait for phones)
- `ROTATION_90`: Sideways (landscape for phones)
- `ROTATION_180`: Upside-down (if device allows)
- `ROTATION_270`: Sideways in the opposite direction
- Many devices return `ROTATION_90` or `ROTATION_270` regardless of clockwise or counterclockwise rotation

Example: Handle device rotation (1)

Use getRotation() with remapCoordinateSystem():

```
float[] rotationMatrix = new float[9];
boolean rotationOK =
        SensorManager.getRotationMatrix(rotationMatrix,
                null, mAccelerometerData, mMagnetometerData);
// Remap matrix based on current device/activity rotation.
float[] rotationMatrixAdjusted = new float[9];
```

Example: Handle device rotation (2)

```
switch (mDisplay.getRotation()) {
case Surface.ROTATION_0:
    rotationMatrixAdjusted = rotationMatrix.clone();
    break;
case Surface.ROTATION_90:
    SensorManager.remapCoordinateSystem(rotationMatrix,
            SensorManager.AXIS_Y, SensorManager.AXIS_MINUS_X,
            rotationMatrixAdjusted);
    Break;
// Rotation_180, Rotation_270 ...
```

Example: Handle device rotation (3)

```
    // Rotation_180, Rotation_270
    case Surface.ROTATION_180:
        SensorManager.remapCoordinateSystem(rotationMatrix,
                SensorManager.AXIS_MINUS_X,
                SensorManager.AXIS_MINUS_Y, rotationMatrixAdjusted);
        break;
    case Surface.ROTATION_270:
        SensorManager.remapCoordinateSystem(rotationMatrix,
                SensorManager.AXIS_MINUS_Y, SensorManager.AXIS_X,
                rotationMatrixAdjusted);
        break;
}
```

Using motion sensors

Monitor device motion such as tilt, shake, rotation, swing

Motion sensors

The movement is usually a reflection of :

- Direct user input relative to device/app (steering car in game, etc.)
- Device motion relative to Earth (device is with you while you are driving)
    - Motion sensors are used with other sensors to determine device position relative to Earth

Accelerometer

- `TYPE_ACCELEROMETER` measures acceleration along 3 device axes (*x*, *y*, *z*) including gravity
- Acceleration without gravity: use `TYPE_LINEAR_ACCELERATION`
- Force of gravity without acceleration: use `TYPE_GRAVITY`
- `TYPE_GYROSCOPE` measures rate of rotation (radians/second)
- For calculations see `SensorEvent` `values`

Accelerometer event data

Event data	Description	Units
`SensorEvent.values[0]`	Acceleration force along *x*-axis, including gravity	m/s^2
`SensorEvent.values[1]`	Acceleration force along *y*-axis, including gravity	m/s^2
`SensorEvent.values[2]`	Acceleration force along *z*-axis, including gravity	m/s^2

Gravity event data

Event data	Description	Units
`SensorEvent.values[0]`	Gravity along *x*-axis	m/s^2
`SensorEvent.values[1]`	Gravity along *y*-axis	m/s^2
`SensorEvent.values[2]`	gravity along *z*-axis	m/s^2

Rotation-vector sensor

- `TYPE_ROTATION_VECTOR` provides orientation with respect to Earth coordinated as unit quaternion
- Software sensor that integrates data from accelerometer, magnetometer, and gyroscope (if available)
- Efficient and accurate way to determine device orientation
- For calculations see `SensorEvent` `values`

Step counter and step detector

- `TYPE_STEP_COUNTER` measures user steps since last reboot
- To preserve battery use `JobScheduler` to retrieve current value from step-counter at specific interval
- `TYPE_STEP_DETECTOR`: hardware sensor that triggers event for each step
- **Example:** See the BatchStepSensor sample app

Using position sensors

Determine device physical position on Earth

Geomagnetic (magnetometer)

- `TYPE_MAGNETIC_FIELD` measures strength of magnetic fields around device on each of 3 axes (*x*, *y*, *z*), including Earth magnetic field
- Units are in microtesla (uT)
- Find device position with respect to external world (compass)

Orientation

- `TYPE_ORIENTATION` deprecated in API 8

- For accurate device orientation *(choose one)*:
  - Use `getRotationMatrix()` and `getOrientation()`, or
  - Use rotation-vector sensor with `TYPE_ROTATION_VECTOR`

What's next?

- Concept chapter: 3.2 Motion and position sensors
- Practical: 3.2 Working with sensor-based orientation

END