JÖNKÖPING UNIVERSITY

School of Engineering

ANDROID SENSORS

Peter Larsson-Green

Jönköping University

Spring 2020

SENSORS

Sensor = object collecting data from its surroundings.

- Android supports many different type of sensor.
  - Temperature - The ambient room temperature.
  - Light - The ambient light level.
  - Pressure - The ambient air pressure.
  - Humidity - The ambient humidity.

SENSORS

Sensor = object collecting data from its surroundings.

- Android supports many different type of sensor.
  - Accelerometer - Acceleration force (including gravity) in 3 directions.
  - Gravity - Gravitational force in 3 directions.
  - Linear Acceleration - Acceleration force (excluding gravity) in 3 directions.
  - Gyroscope - Rotational speed around 3 axes.
- Sensors can be implemented either as hardware or software.
- All Android devices do not support all sensors.

SENSOR SUPPORT TABLE

Sensor	Android 4.0	Android 2.3	Android 2.2	Android 1.5
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	n/a
TYPE_LIGHT	Yes	Yes	Yes	Yes
TYPE_LINEAR_ACCELERATION	Yes	Yes	n/a	n/a
TYPE_MAGNETIC_FIELD	Yes	Yes	Yes	Yes
TYPE_ORIENTATION	Deprecated	Deprecated	Deprecated	Yes
TYPE_PRESSURE	Yes	Yes	n/a	n/a
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	Deprecated	Yes	Yes	Yes

USING SENSORS

```
SensorManager sensorManager = (SensorManager)
          aContext.getSystemService(Context.SENSOR_SERVICE);
Sensor sensor = sensorManager.
                        getDefaultSensor(Sensor.TYPE_XXX);
SensorEventListener listener = new SensorEventListener(){
  @Override
  public void onSensorChanged(SensorEvent event){
  }
  @Override
  public void onAccuracyChanged(Sensor sensor, int accuracy){
  }
};
```

```
SENSOR_STATUS_NO_CONTACT
SENSOR_STATUS_UNRELIABLE
SENSOR_STATUS_ACCURACY_LOW
SENSOR_STATUS_ACCURACY_MEDIUM
SENSOR_STATUS_ACCURACY_HIGH
```

USING SENSORS

```
SensorEventListener listener = new SensorEventListener(){
  protected float[] myValues = new float[3];
  @Override
  public void onSensorChanged(SensorEvent event){
    Sensor sensor = event.sensor;
    float[] values = event.values;
    System.arraycopy(event.values, 0,
                     myValues, 0,
                     event.values.length);
  }
};
```

USING SENSORS

To start listen for events:

```
sensorManager.registerListener(
    listener,
    sensor,
    SensorManager.SENSOR_DELAY_XXX
);
```

```
SENSOR_DELAY_FASTEST
SENSOR_DELAY_GAME
SENSOR_DELAY_NORMAL
SENSOR_DELAY_UI
```

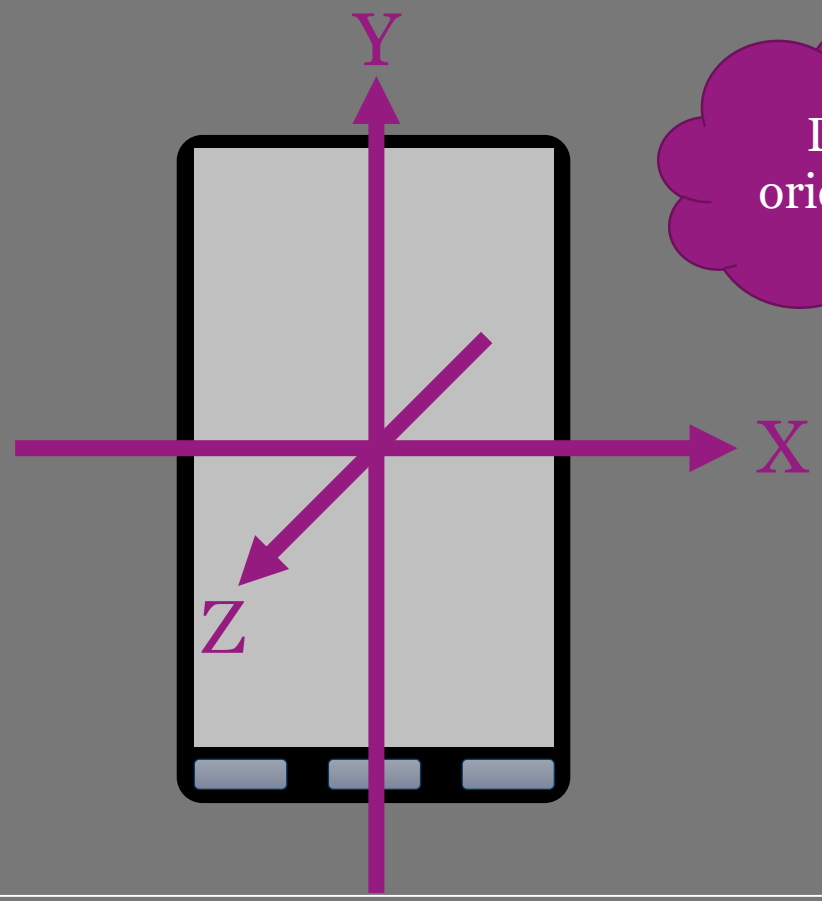
To stop listen for events:

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

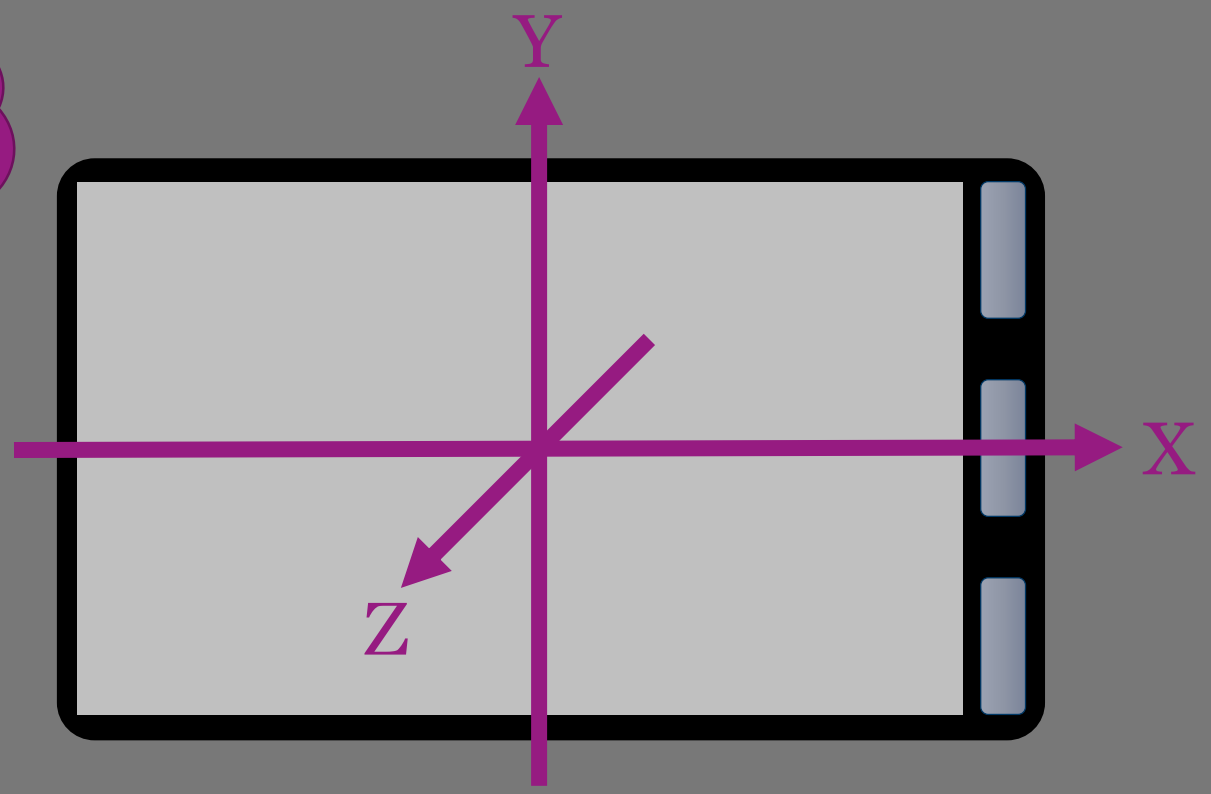
Documentation: "*Note that the system will not disable sensors automatically when the screen turns off*".

COORDINATE SYSTEM FOR MOTION

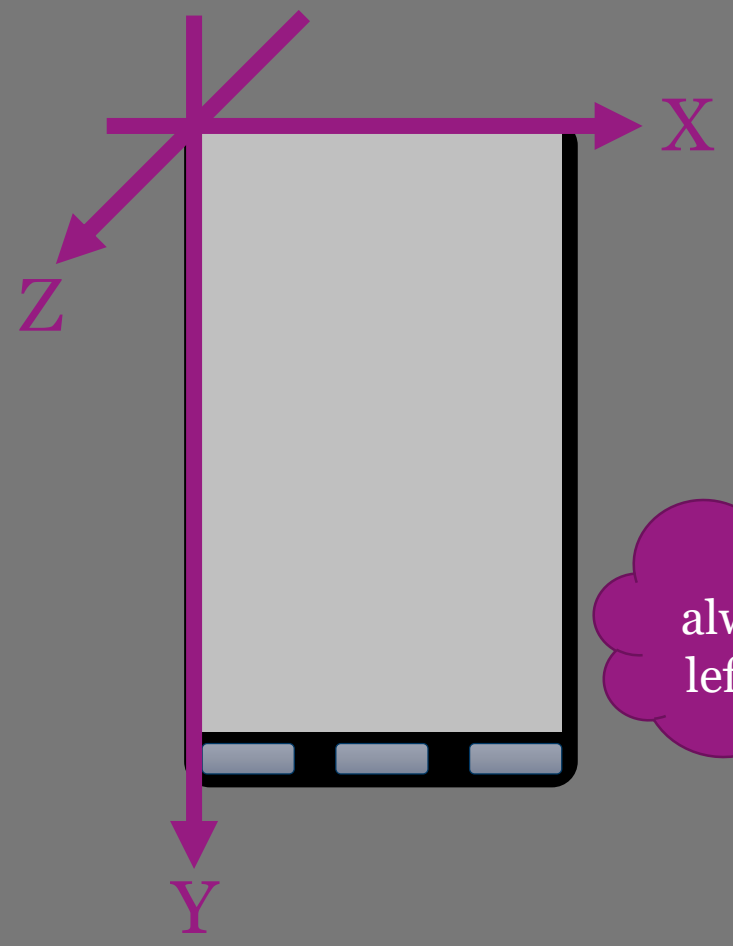
The device's frame of reference.



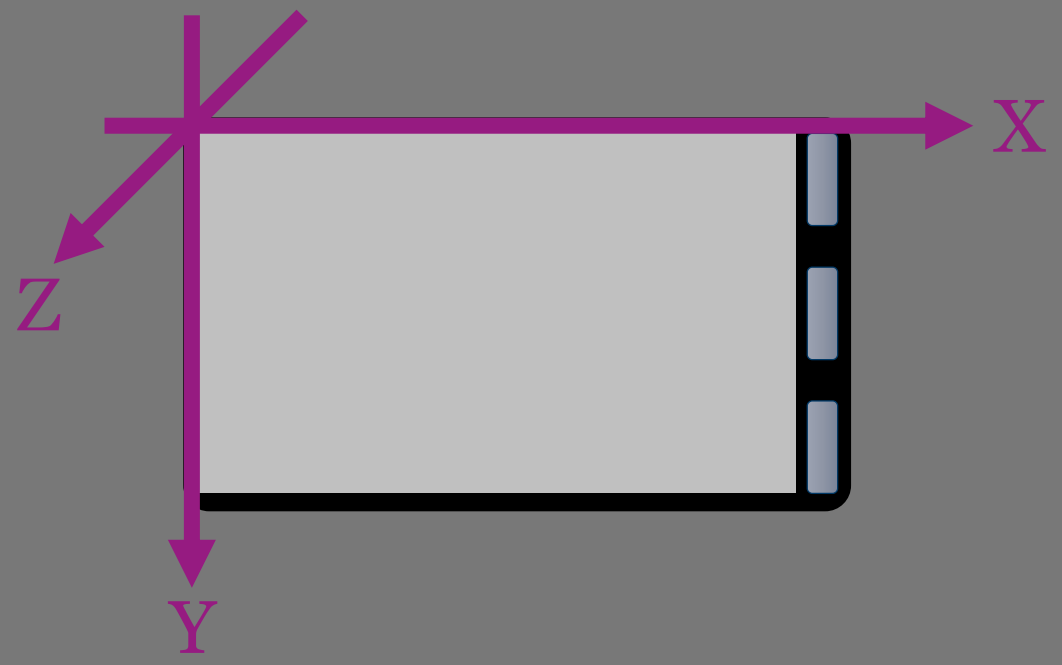
Default orientation!



COORDINATE SYSTEM FOR UI



Origin always top-left corner.



Unless locked in portrait mode.

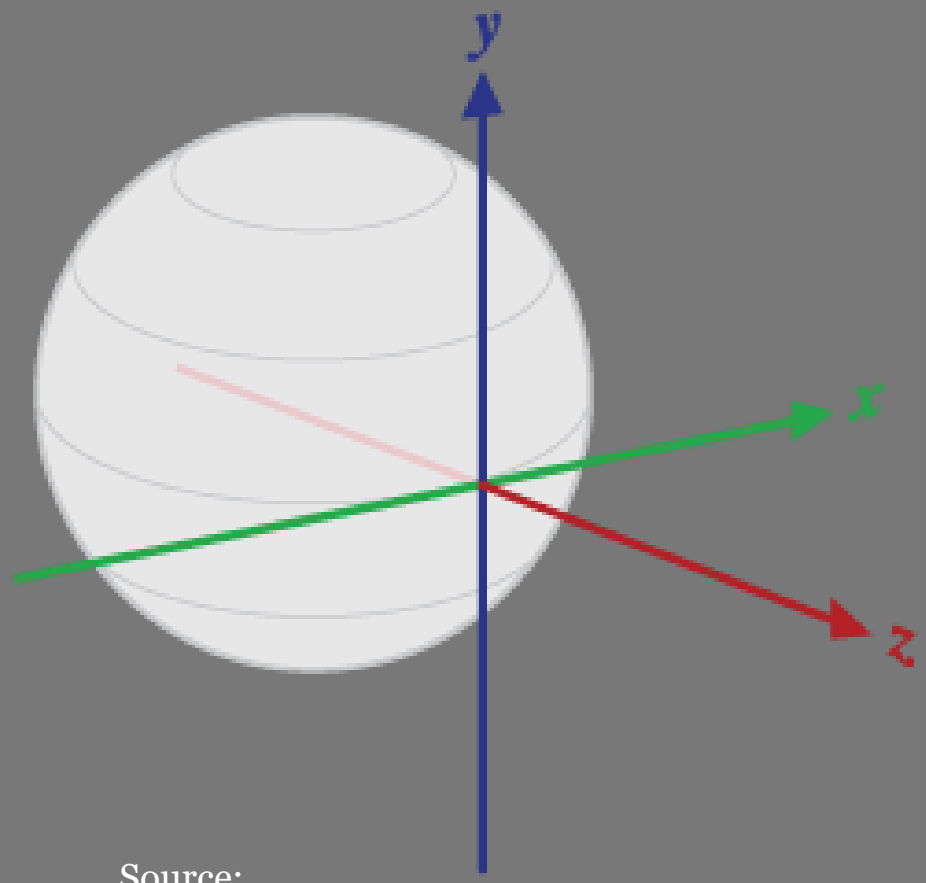
FINDING CURRENT ORIENTATION

```
int rotation = anActivity.getWindowManager().
                                getDefaultDisplay().getRotation();
switch(rotation){
  case Surface.ROTATION_0:
    // Default orientation is used.
  break;
  case Surface.ROTATION_90:
    // Default orientation + 90 degrees.
  break;
  // ...
}
```

COORDINATE SYSTEM FOR EARTH

The world's frame of reference.

- Z points towards the sky.
- Y points towards the magnetic North Pole.
- X points towards the East(-ish).

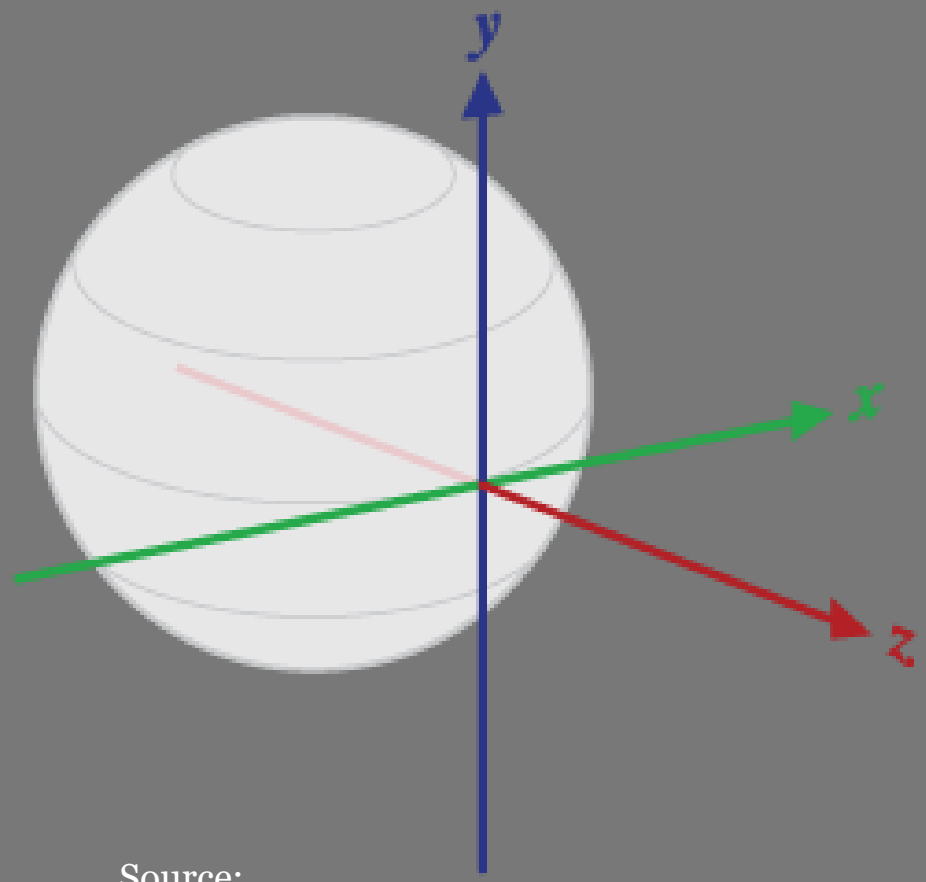


Source: https://developer.android.com/reference/android/hardware/SensorManager.html

COORDINATE SYSTEM FOR EARTH

```
SensorManager.getRotationMatrix(
  float[9] R,
  float[9] I,
  accelerometerValues,
  geomagneticValues
);
```

Computes the inclination matrix **I** as well as the rotation matrix **R** transforming a vector from the device coordinate system to the world's coordinate system.



Source: https://developer.android.com/reference/android/hardware/SensorManager.html

GETTING THE ORIENTATION

```
SensorManager.getOrientation(
  R,
  float[3] rotations
);
```

Computes the device's orientation based on the rotation matrix.

- `rotations[0]` - Rotation around the Z axis.
- `rotations[1]` - Rotation around the X axis.
- `rotations[2]` - Rotation around the Y axis.

REMAPPING SENSOR DATA

```
SensorManager.remapCoordinateSystem(
  oldR, SensorManager.AXIS_Y, SensorManager.AXIS_Z,
  float[9] newR
);
```

- The third axis?
  - The system figures it out (orthonormal system).

The new X axis.

The new Y axis.