

Mobile Computing

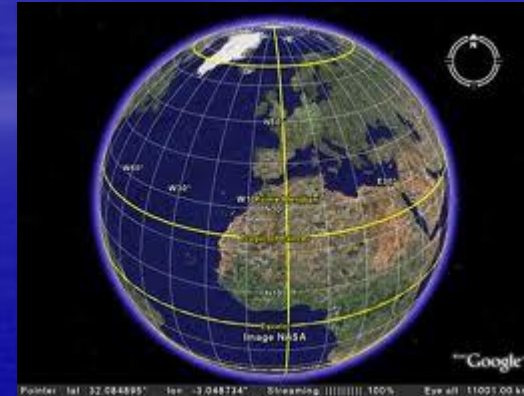
Sensors

Location

❖ Where in the world am I ?

● Earth surface point

- Latitude and longitude (°)
- Altitude (m)
- Accuracy (m)



❖ Status of movement

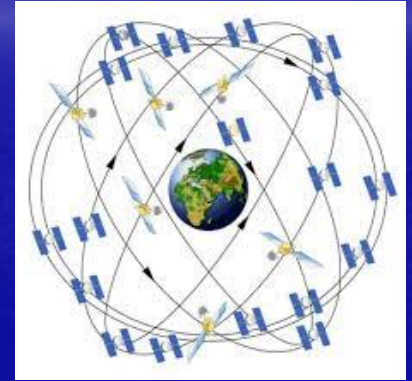
- Speed (m/s)
- Bearing (° from North)



Location providers

❖ A device may have several providers

- Location can be obtained directly from satellites (GPS)
- Or can be derived from
 - Wi-fi access points information
 - Mobile communication tower's location



❖ They are condensed in the providers

- GPS_PROVIDER (“gps”)
- NETWORK_PROVIDER (“network”)
- Also usually exists a PASSIVE_PROVIDER (another app)

❖ GPS

- Accurate, more info, more power consumption, more delay, needs line of sight to satellites (weak signals)

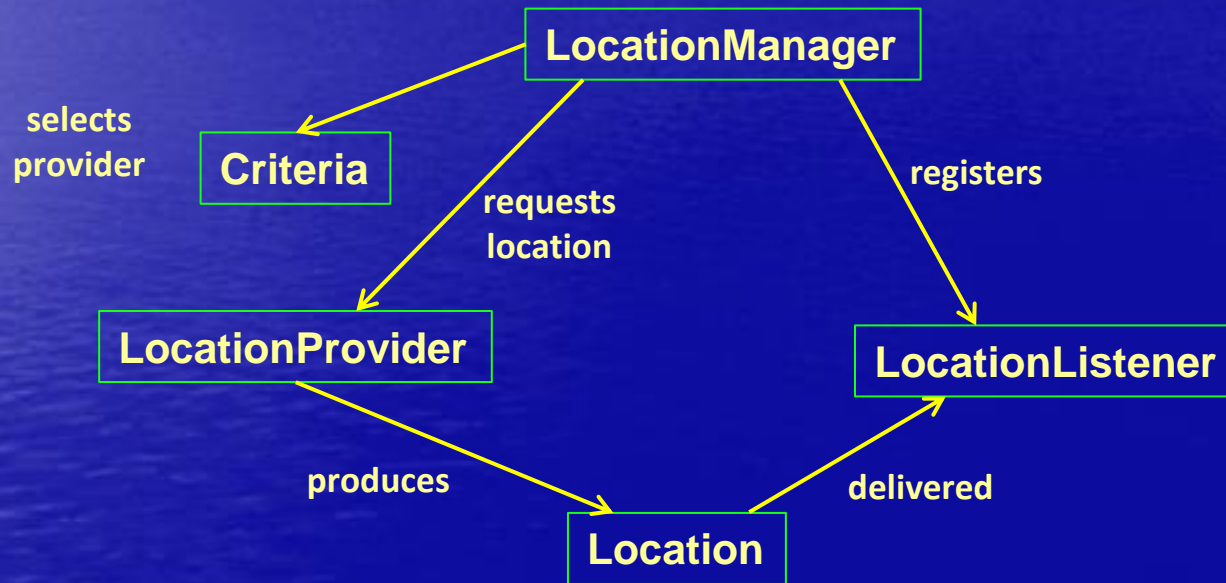
❖ Network

- Less accurate, less consumption, less delay, interiors

Android classes and permissions

❖ Android manifest must declare permissions

- **ACCESS_COARSE_LOCATION** and/or
- **ACCESS_FINE_LOCATION**



LocationListener

Interface declaring methods (callbacks) where information is delivered after requesting a location

```
public interface LocationListener {  
    void onLocationChanged(Location location);  
    void onProviderDisabled(String provider);  
    void onProviderEnabled(String provider);  
    void onStatusChanged(String provider, int status, Bundle extras);  
}
```

not called in
API 29

OUT_OF_SERVICE
TEMPORARILY_UNAVAILABLE
AVAILABLE

satellites:
nr. of satellites in gps

LocationManager

System service for requesting locations and selecting a provider

Obtained in an Activity:

```
locationManager = (LocationManager) getSystemService(LOCATION_SERVICE);
```

Locations should be requested only when the activity is active and cancelled when it stops being in foreground. So requests can be done in the `onResume()` callback of the activity, from the LocationManager object, with:

minimum time or distance
for a new location

selection of
a provider

```
requestLocationUpdates(long minTime, float minDistance, Criteria criteria,  
LocationListener listener, Looper looper)
```

(there are more overrides)

class implementing
the listener

null for the
current thread

In `onPause()` we should cancel the request and stop the provider:

```
removeUpdates(LocationListener listener)
```

It is also possible to request a single location or an alert of proximity to a given location expressed in latitude or longitude.

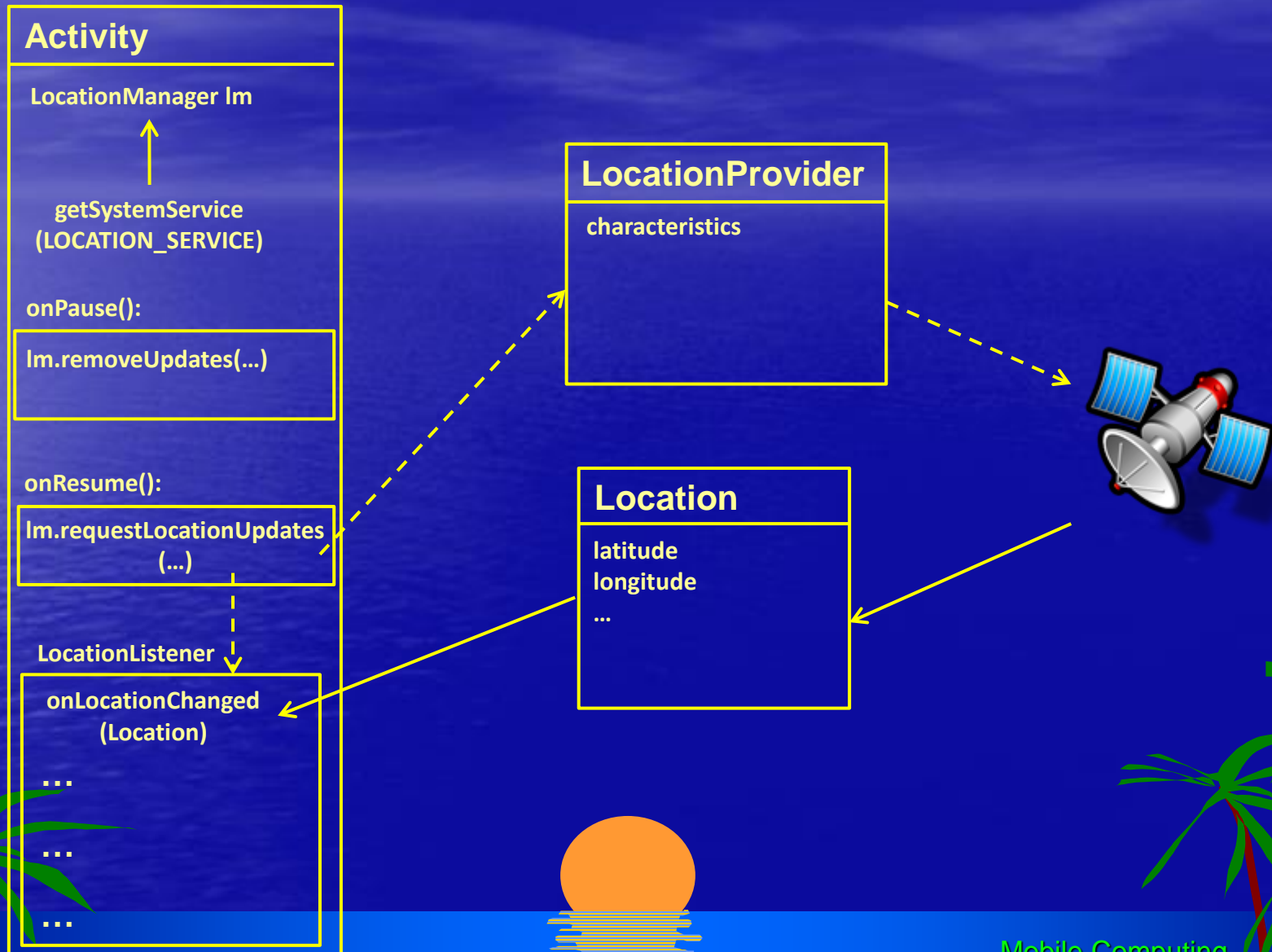
```
requestSingleUpdate(...) and addProximityAlert(...)
```

Location

❖ Locations are delivered to **LocationListener**

- They bring latitude and longitude
- The provider that has generated it
- The time it was generated
- If the provider have the information, it can also contain
 - The altitude of the location
 - The accuracy of the location
 - The bearing if the device is moving
 - The speed also if it is moving
 - The number of satellites used to obtain the location
- **Some convenience methods allow**
 - To know the bearing from this location to another (geodesic)
 - The distance between locations (along a geodesic)
 - Obtain a convenient String representation

Receiving location information



Sensors

❖ Android has support for several sensors and a common API to get their measurements

- **Sensors available include**

- movement sensors
- orientation sensors
- environment sensors

- **Sensors can be divided into**

- physical sensors giving actual measurements of some quantity
- synthesized sensors fusing and processing the measurements of other physical sensors to calculate another quantity

- **Sensors can be characterized by**

- Range and resolution (minimum, maximum and step value)
- Rate of measurement (nr. of measurements per time unit)
- power consumption



Sensor types supported

Android defines a constant for each of the sensor types supported by the operating system, in the Sensor class. But each device may have only a few of those sensors.

Defined types:

physical

synthetized

movement:

TYPE_ACCELEROMETER (3D)

TYPE_GYROSCOPE (3D)

orientation:

TYPE_MAGNETIC_FIELD (3D)

environment:

TYPE_AMBIENT_TEMPERATURE

TYPE_LIGHT

TYPE_PRESSURE

TYPE_PROXIMITY

TYPE_RELATIVE_HUMIDITY

TYPE_GRAVITY (3D)

TYPE_LINEAR_ACCELERATION (3D)

TYPE_ORIENTATION (3D)

TYPE_ROTATION_VECTOR (3 or 4D)

TYPE_SIGNIFICANT_MOTION (Trigger)

For getting a list of all sensors present on a device, there is the constant:

TYPE_ALL

List of sensors are acquired from the SensorManager calling `getSensorList(type)`

Sensors on a device

It's possible for a device to have more than one sensor of the same type. It's uncommon for physical sensors, but can happen with synthesized sensors (more than one implementation).

A screenshot of an Android phone's sensor list. The status bar at the top shows a USB icon, signal strength, battery level, and the time 11:32. The list is titled "Sensor List" and contains the following entries: AK8975C 3-axis Magnetic field sensor, iNemoEngine Orientation sensor, CM36651 Light sensor, CM36651 Proximity sensor, LSM330DLC Gyroscope sensor, iNemoEngine Gravity sensor, iNemoEngine Linear Acceleration sensor, iNemoEngine Rotation_Vector sensor, LPS331AP Pressure Sensor, Rotation Vector Sensor, Gravity Sensor, and Linear Acceleration Sensor.

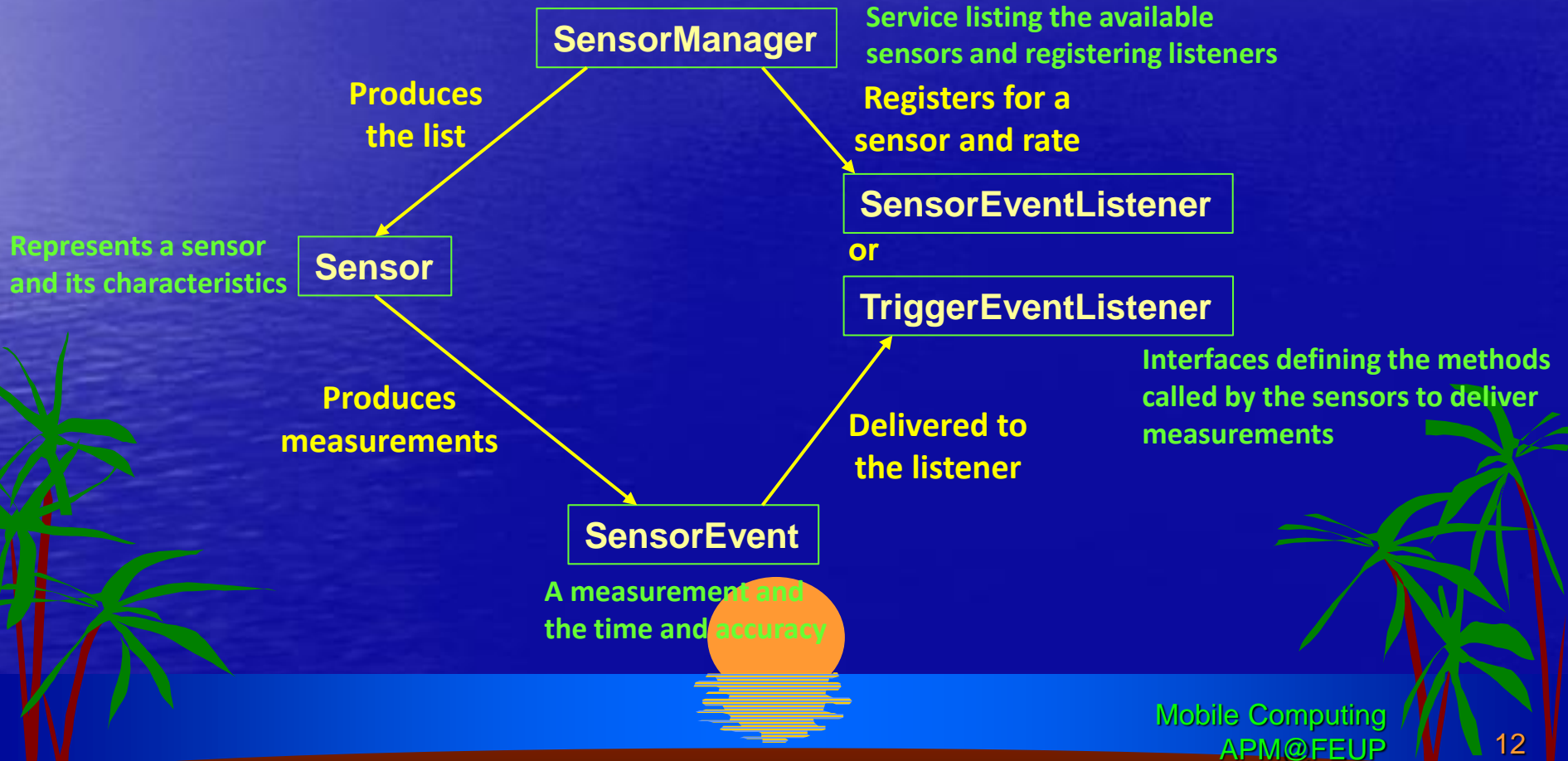
Sensor List
AK8975C 3-axis Magnetic field sensor
iNemoEngine Orientation sensor
CM36651 Light sensor
CM36651 Proximity sensor
LSM330DLC Gyroscope sensor
iNemoEngine Gravity sensor
iNemoEngine Linear Acceleration sensor
iNemoEngine Rotation_Vector sensor
LPS331AP Pressure Sensor
Rotation Vector Sensor
Gravity Sensor
Linear Acceleration Sensor

When an application requests the sensors of a certain type (with `getSensorList(type)`), the system returns an array of sensors.

Sensor API classes

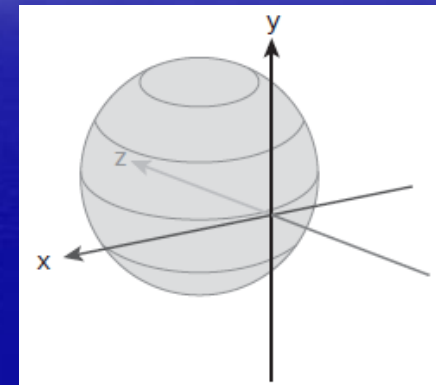
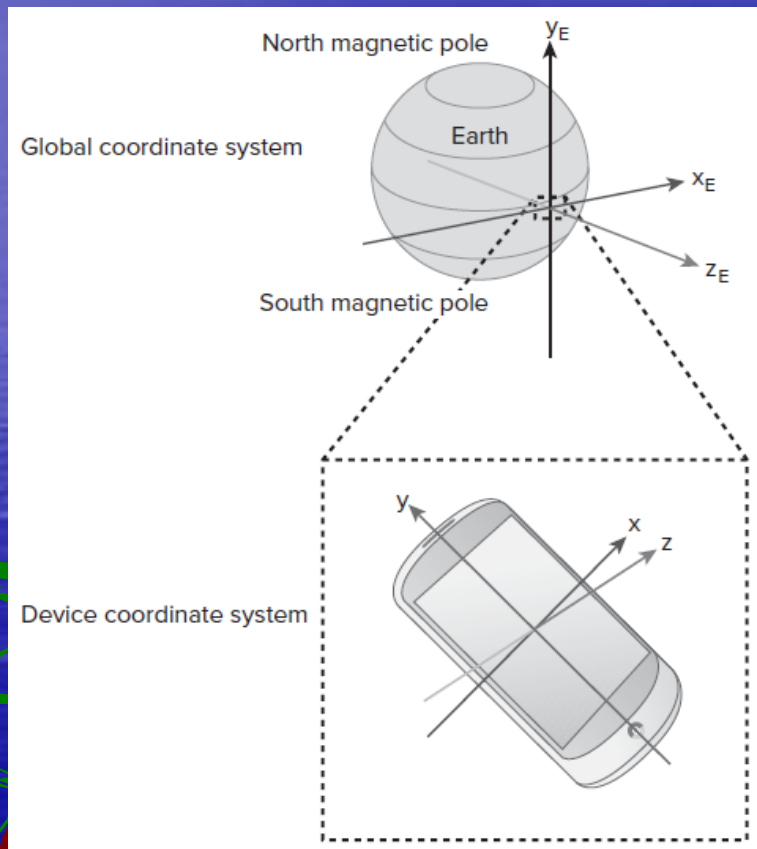
The **SensorManager** is the starting point and can be obtained from the Activity with:

```
(SensorManager) getSystemService(SENSOR_SERVICE)
```



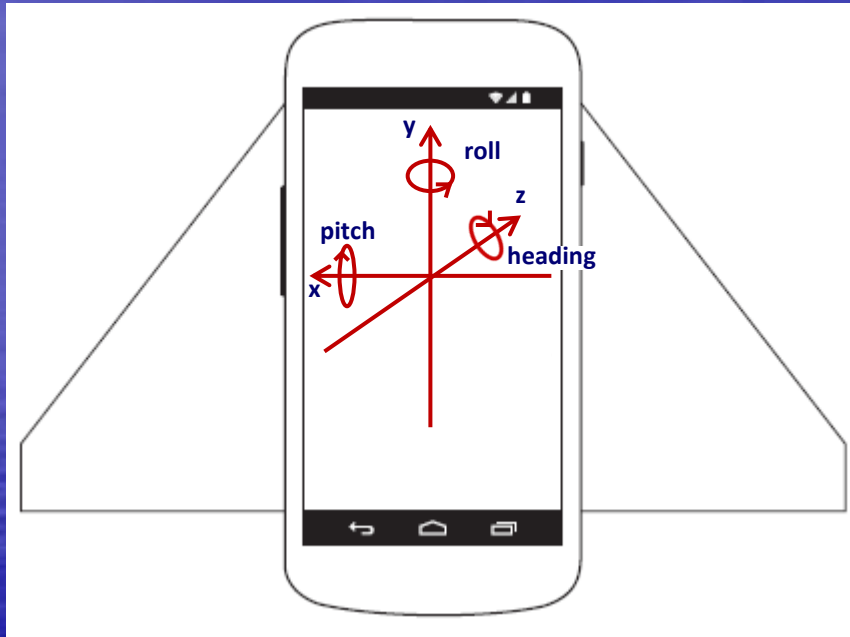
Movement and orientation

Movement quantities are presented in the device coordinate system. Orientation measures are in the earth coordinate system (except TYPE_ORIENTATION and getOrientation() measurements).



**TYPE_ORIENTATION
and getOrientation()
measurements
earth coordinate system**

Orientation measurements



Heading – 0° to 360°

Pitch – -180° to 180°

Roll – -90° to 90°

SensorManager

- ❖ Activities get it from the system
- ❖ It knows the sensors available on the device
 - We can get a list of all or of a single type of sensors
 - It's possible to have more than one sensor of a given type (specially of fusion/synthesized sensors)
- ❖ It can register the **SensorEventListener** for one or more sensors
- ❖ It defines some measure transformation methods
 - **getRotationMatrixFromVector()**
 - uses the ROTATION_VECTOR sensor and computes a rotation matrix
 - **getRotationMatrix()**
 - computes Inclination and Rotation matrices from gravity and geomagnetic fields
 - **getInclination()** (from the Inclination matrix)
 - **getOrientation()** (from the Rotation matrix)
 - **getAltitude()**
 - From the atmospheric pressure here and at sea level

SensorEventListener

Interface declaring methods (callbacks) where measurements are delivered after registering it for a sensor.

Measurements are represented by an instance of SensorEvent

```
public interface SensorEventListener {  
    void onSensorChanged(SensorEvent event);  
    void onAccuracyChanged(Sensor sensor, int accuracy);  
}
```

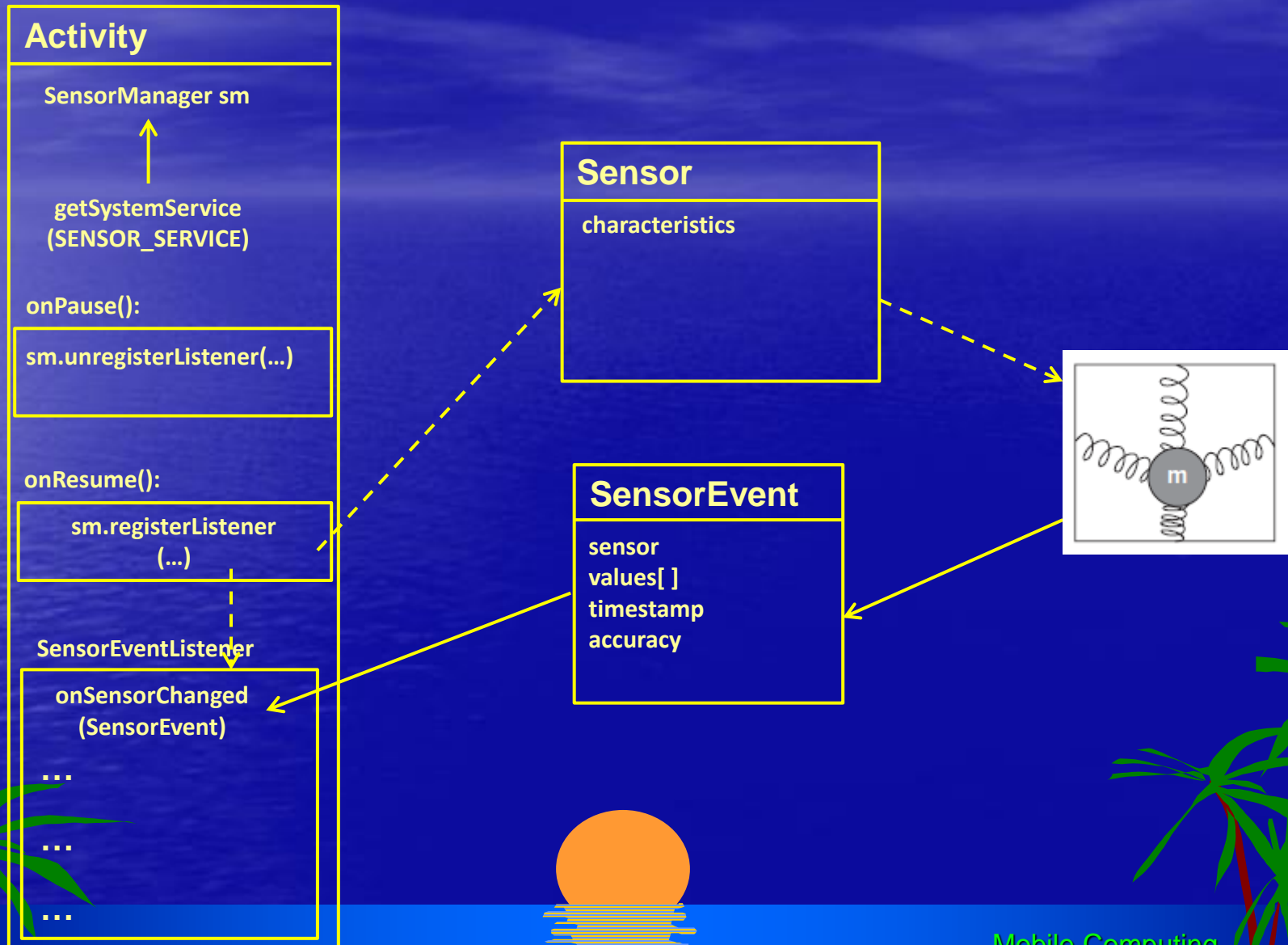
sensor
values[]
timestamp
accuracy

For sensors producing an event at a time (like the SIGNIFICANT_MOTION detector) use the abstract class:

```
public abstract class TriggerEventListener {  
    public void onTrigger(TriggerEvent event);  
}
```

SENSOR_STATUS_ACCURACY_HIGH
SENSOR_STATUS_ACCURACY_MEDIUM
SENSOR_STATUS_ACCURACY_LOW
SENSOR_STATUS_ACCURACY_UNRELIABLE

Receiving sensor measurements



Noise and signal processing

- ❖ Many sensors produce several kinds of noise
 - High frequency variations with significant amplitudes
 - Low frequency deviations (drifts)
- ❖ Some simple frequency domain filters are useful
 - Low pass filters
 - Weighted smoothing
 - Simple moving average
 - Simple moving median
 - High pass filters
 - Inverse low pass filter
 - Band pass filters and its inverse
 - Simultaneous low and high pass
 - Kalman filters

