SENSORS

Location service

- Mobile applications can benefit from being location-aware
- This mean to allow application to determine and manipulate location
- For example:
  - find stores nead my current location
  - Direct the user from the current location to a specific place

- Geofence
  - Perform an action when a user enters or leaves the geofence area

Location

- Represents a position on the Earth
- A location instance consists of:
- Latitude, Longitude,Timestamp (optionally, accuracy, altitude, etc)

Location provider

- Represents a location data source

- GPS satellites
- Network provider
  - Cell phone towers
  - Wi-fi access points
- Passive provider
  - Location from other application

Requires
android.permission.ACCESS_COARSE_LOCATION
android.permission.ACCESS_FINE_LOCATION

Provider tradeoffs

- GPS – expensive, accurate, slower, outdoors
- Network - cheapear, less accurate, faster, availability varies
- Passive – cheapest, fastest, may be less accurate, not always available

LocationManager

- System Service for accessing location data
- GetSystemService(Context.LOCATION_SERVICE)
- After that register to get last known user location, location updates, or receive intents when the devide nears or moving away from a geographic region

LocationListener

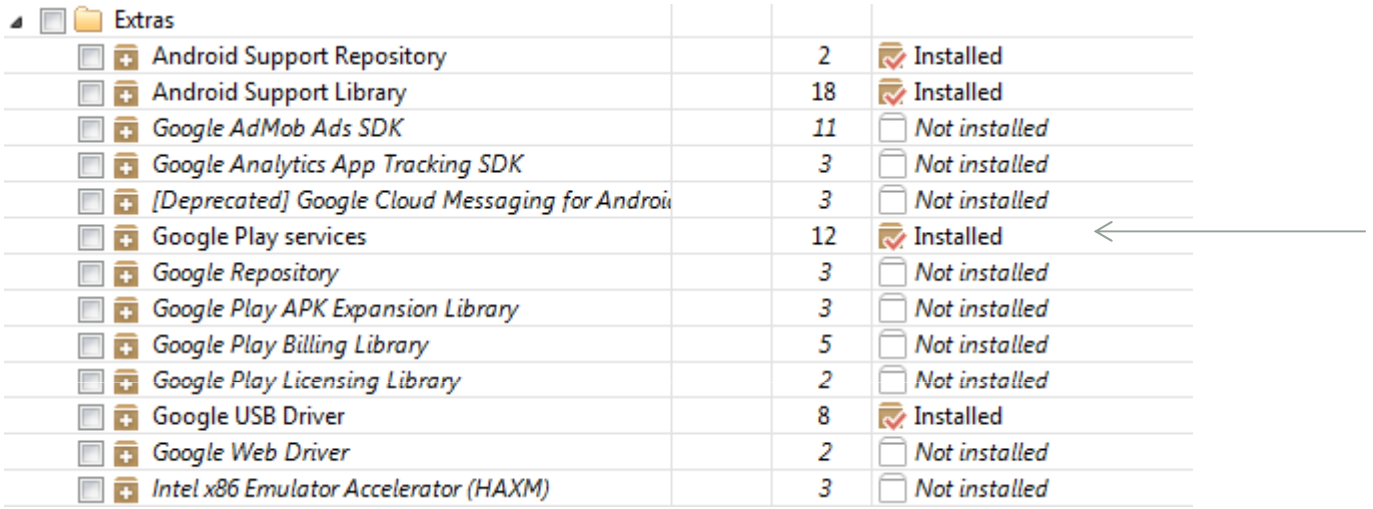
- Interface defines callback methods called when location changes or locationProvider status changes
- It includes the following method

- Void onLocationChanged(Location location)
- Void onProviderDisabled(String provider)
- Void on ProviderEnabled (String provider)

New API for location

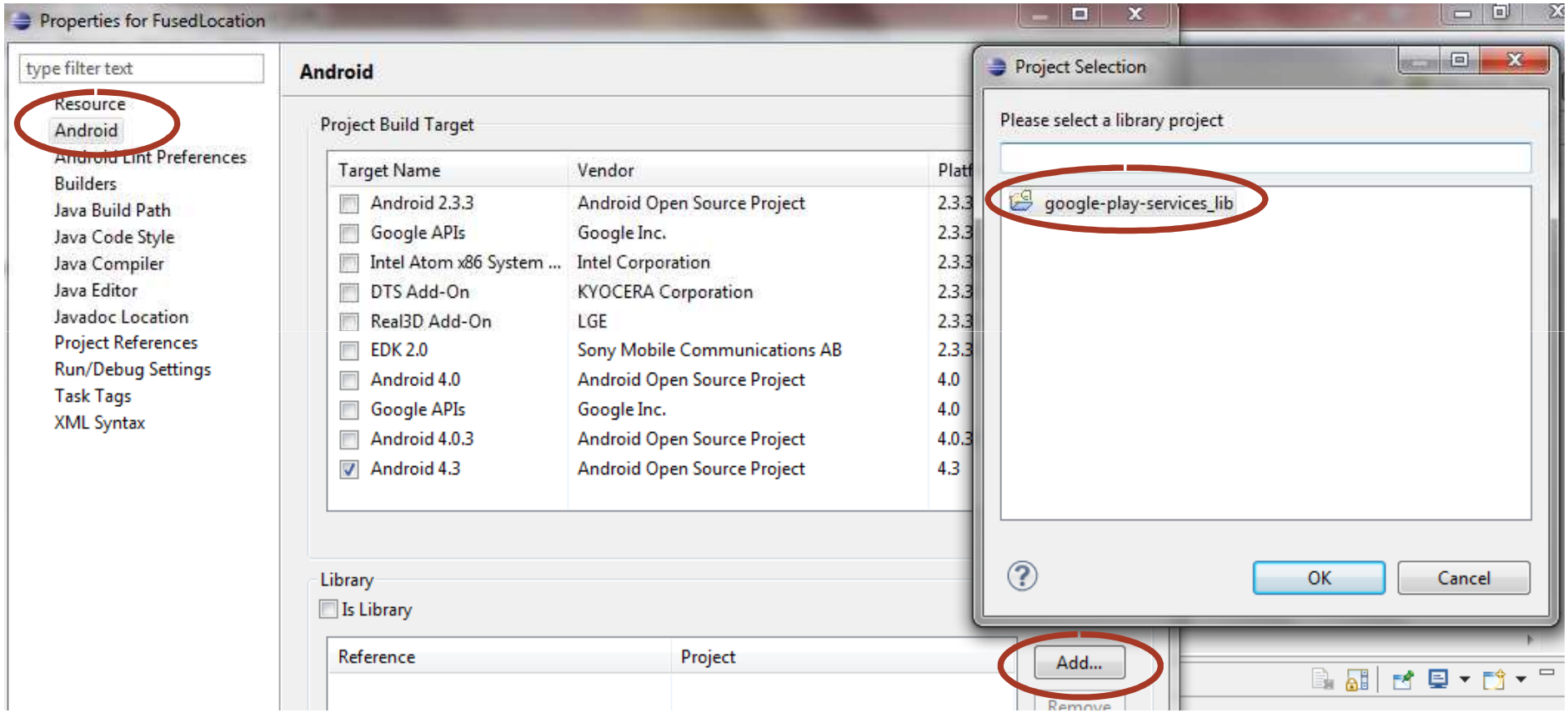
- Fused Location Provider
  - Better precision, low energy
- Geofencing
  - Allows for making virtual limits and signal when enter or leave a specif area
- Activity recognition
  - Recognize movements without GPS (i.e., using accelerometer)

Example. First step, add library



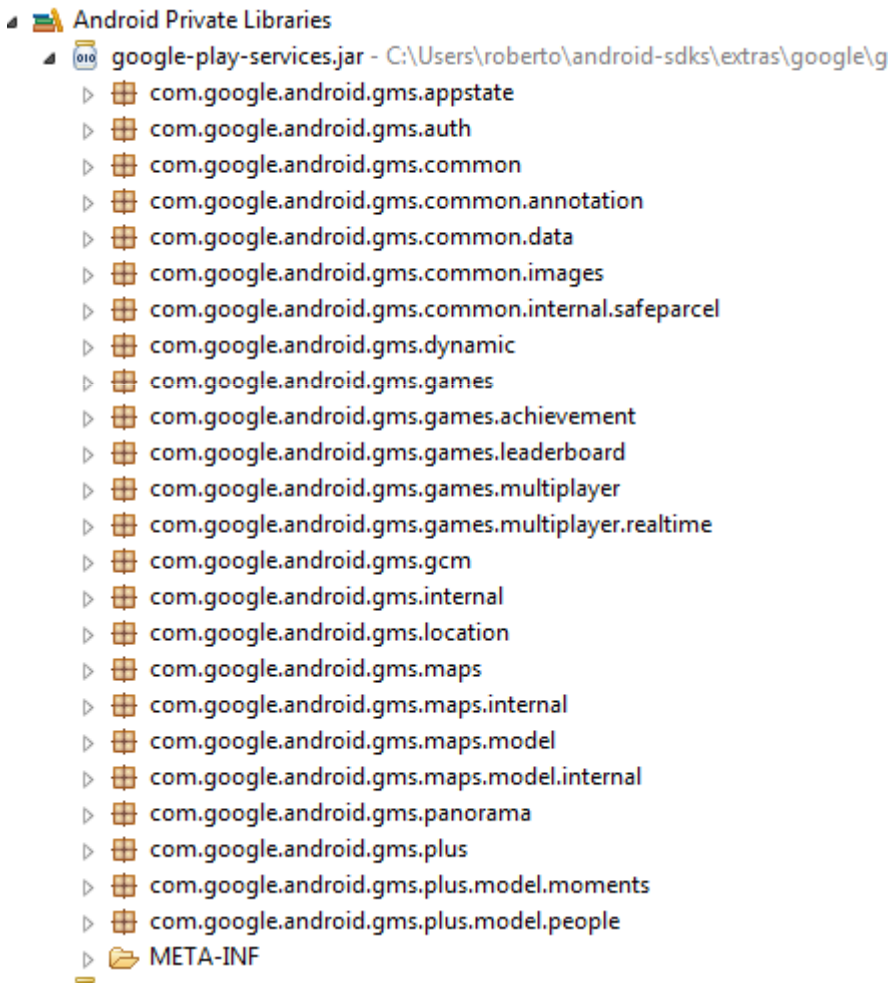
- See documentation about how to install the library

Example. 2 step, import library



- Android SDK manager
- Import, google play

Example. Library content



Example. Getting location once

```
package com.example.fusedlocation;

import com.google.android.gms.common.ConnectionResult;
import com.google.android.gms.common.GooglePlayServicesClient;
import com.google.android.gms.common.GooglePlayServicesUtil;
import com.google.android.gms.location.LocationClient;

import android.app.Activity;
import android.location.Location;
import android.location.LocationListener;
import android.os.Bundle;
import android.view.Menu;
import android.widget.Toast;

public class MainActivity extends Activity implements
GooglePlayServicesClient.ConnectionCallbacks,
GooglePlayServicesClient.OnConnectionFailedListener,
LocationListener


@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    int resp =GooglePlayServicesUtil.isGooglePlayServicesAvailable(this);

    if(resp == ConnectionResult.SUCCESS){
           locationclient = new LocationClient(this,this,this);
           locationclient.connect();
     }


}
```

Add permission to the manifest file

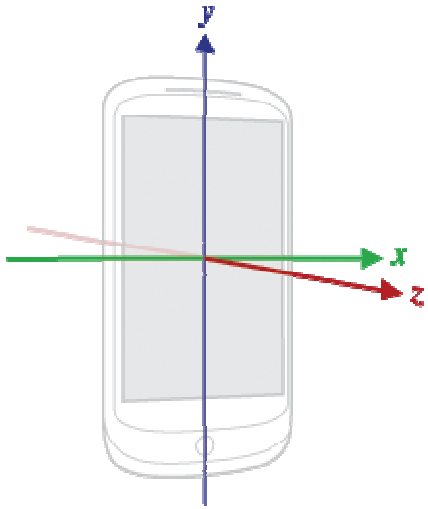
Example. Getting location periodically

```
int resp =GooglePlayServicesUtil.isGooglePlayServicesAvailable(this);

if(resp == ConnectionResult.SUCCESS){
       locationclient = new LocationClient(this,this,this);
       locationclient.connect();
       locationrequest = LocationRequest.create();
       locationrequest.setInterval(100);


  @Override
  public void onConnected(Bundle arg0) {
      // TODO Auto-generated method stub
      locationclient.requestLocationUpdates(locationrequest, this);
  }

@Override
public void onLocationChanged(Location loc) {
    // TODO Auto-generated method stub
    String msg = "location changed;"+loc.toString();
    Toast.makeText(this,msg,0).show();

}
```

Define a locationRequest
Set parameters
Set request updated in the onConnected Method
Read location in onLocationUpdade method

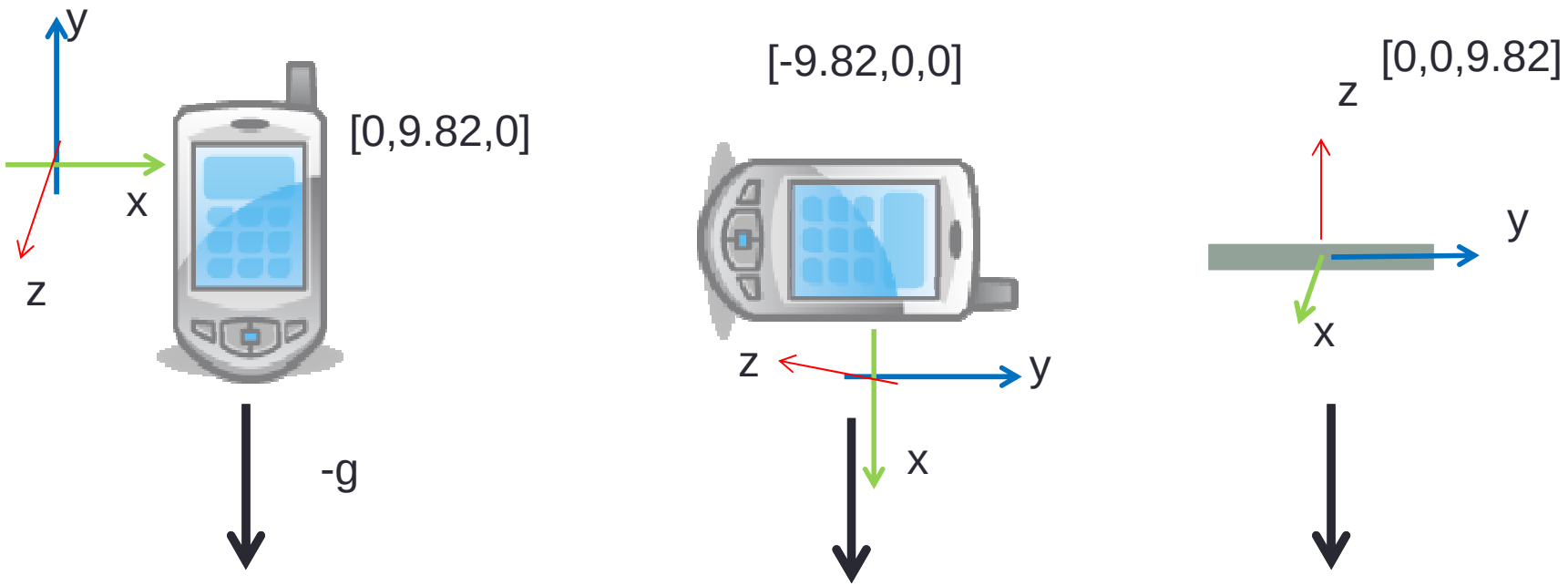
Reference systems

- Android uses two different reference systems
- The first coordinate system is relative to the screen of the phone:
  - X axis: is horizontal and points to the right,
  - Y axis: is vertical and points up
  - Z axis points towards the outside of the front face of the screen.
    - coordinates behind the screen have negative Z values.



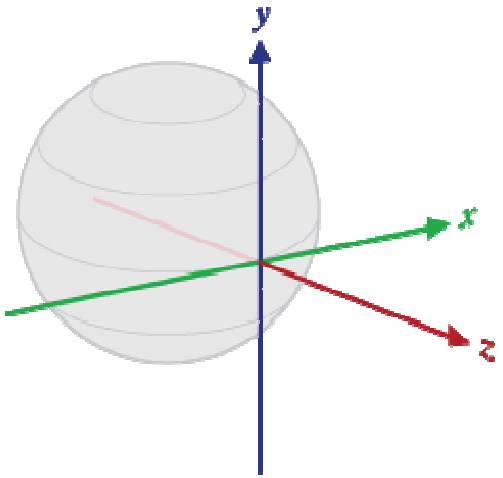
Reference systems

- Readings are done in the phone's system reference
- For example, the accelerometer measures the acceleration applied to the phone minus the force of gravity (~9,82 m/s², depending on the position)
- For example, for a free falling device the acceleration is 0



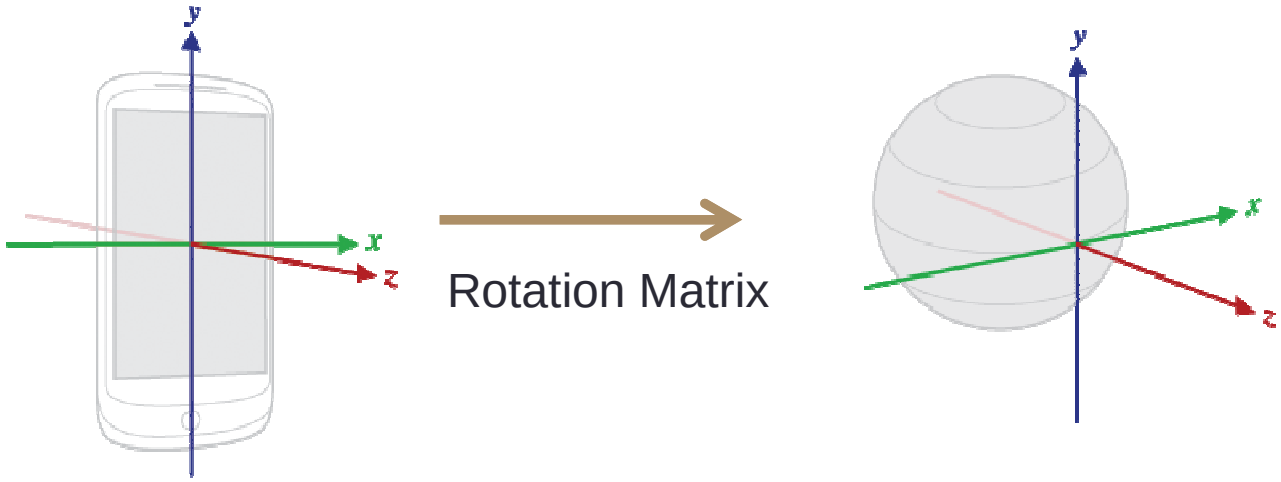
Reference systems

- The other coordinate system is relative to the earth:
  - X is defined as the vector product **YZ** (It is tangential to the ground at the device's current location and roughly points East).
  - Y is tangential to the ground at the device's current location and points towards magnetic north.
  - Z points towards the sky and is perpendicular to the ground.



Coordinate transformation

- In order to express a reading in the earth's coordinate system, the vector should be multiplied by a **rotation matrix**



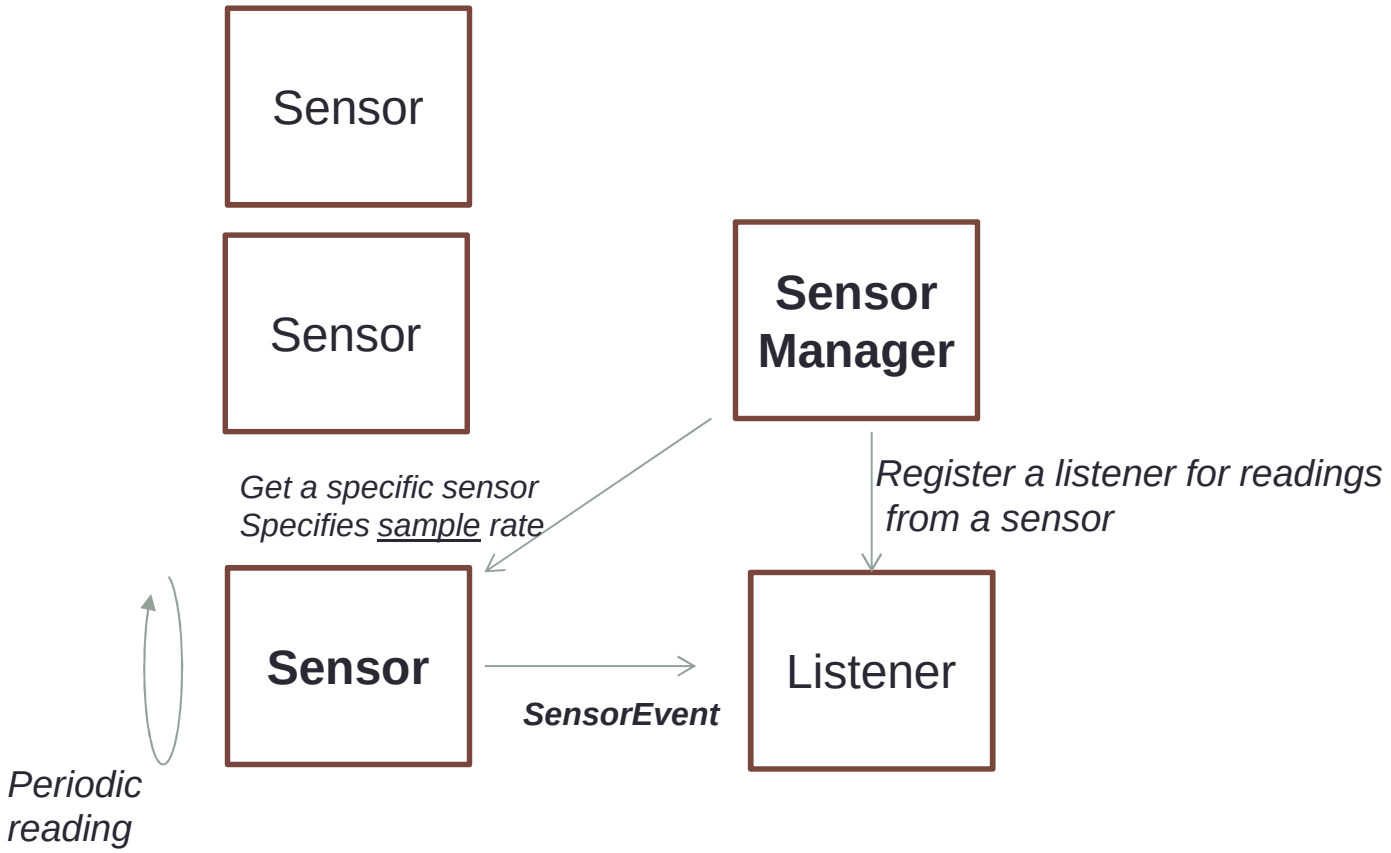
- The rotation matrix is calculated starting from two *reference vectors*:
  - gravity
  - magnetic field
- When the orientation on the device is the same of the earth's one, the matrix is the identity matrix

Sensor types

- An Android devices can have different sensors

int	TYPE_ACCELEROMETER	A constant describing an accelerometer sensor type.
int	TYPE_ALL	A constant describing all sensor types.
int	TYPE_AMBIENT_TEMPERATURE	A constant describing an ambient temperature sensor type
int	TYPE_GRAVITY	A constant describing a gravity sensor type.
int	TYPE_GYROSCOPE	A constant describing a gyroscope sensor type
int	TYPE_LIGHT	A constant describing a light sensor type.
int	TYPE_LINEAR_ACCELERATION	A constant describing a linear acceleration sensor type.
int	TYPE_MAGNETIC_FIELD	A constant describing a magnetic field sensor type.
int	TYPE_ORIENTATION	*This constant was deprecated in API level 8. use* `SensorManager.getOrientation()` *instead.*
int	TYPE_PRESSURE	A constant describing a pressure sensor type
int	TYPE_PROXIMITY	A constant describing a proximity sensor type.
int	TYPE_RELATIVE_HUMIDITY	A constant describing a relative humidity sensor type.
int	TYPE_ROTATION_VECTOR	A constant describing a rotation vector sensor type.
int	TYPE_TEMPERATURE	*This constant was deprecated in API level 14. use* `Sensor.TYPE_AMBIENT_TEMPERATURE` *instead.*

Software architecture



Software architecture

`SensorManager`

You can use this class to create an instance of the sensor service. This class provides various methods for accessing and listing sensors, registering and unregistering sensor event listeners, and acquiring orientation information. This class also provides several sensor constants that are used to report sensor accuracy, set data acquisition rates, and calibrate sensors.

`Sensor`

You can use this class to create an instance of a specific sensor. This class provides various methods that let you determine a sensor's capabilities.

`SensorEvent`

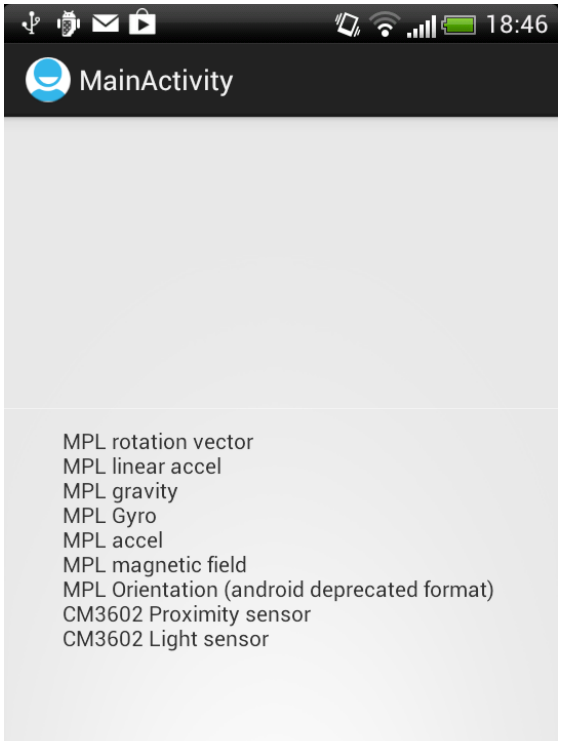
The system uses this class to create a sensor event object, which provides information about a sensor event. A sensor event object includes the following information: the raw sensor data, the type of sensor that generated the event, the accuracy of the data, and the timestamp for the event.

`SensorEventListener`

You can use this interface to create two callback methods that receive notifications (sensor events) when sensor values change or when sensor accuracy changes.

Example: Show the list of sensors

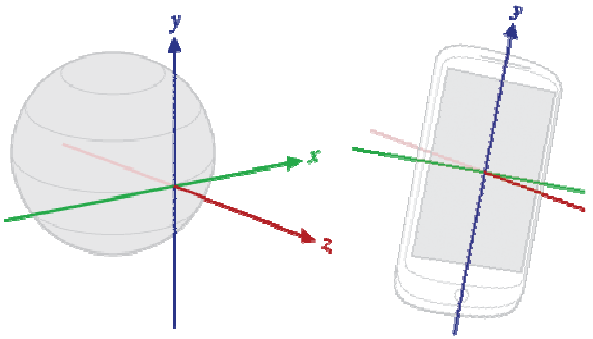
```java
package com.example.sensordemo;

import java.util.Iterator;
import java.util.List;

import android.app.Activity;
import android.hardware.Sensor;
import android.hardware.SensorManager;
import android.os.Bundle;
import android.widget.TextView;

public class MainActivity extends Activity {

    private SensorManager sm = null;
    private List<Sensor> deviceSensors = null;
    private TextView tv=null;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        tv = (TextView)findViewById(R.id.log);
        sm = (SensorManager) getSystemService(SENSOR_SERVICE);
        deviceSensors = sm.getSensorList(Sensor.TYPE_ALL);
        Iterator<Sensor> it = deviceSensors.iterator();
        while (it.hasNext()) tv.setText(tv.getText()+it.next().getName()+"\n");

    }

}
```




```xml
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <TextView
        android:id="@+id/log"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:padding="@dimen/padding_medium"
        tools:context=".MainActivity" />

</RelativeLayout>
```

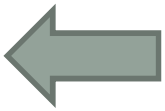
Magnetic Sensor

- Sensor.TYPE_MAGNETIC_FIELD
- Values[3] = the three magnetic field components, in micro-Tesla
- SensorManager’s constants
  - MAGNETIC_FIELD_EARTH_MAX: 60
  - MAGNETIC_FIELD_EARTH_MIN: 30

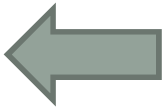


Example: reading magnetic field

```
package com.example.sensordemo2;

import android.app.Activity;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.os.Bundle;
import android.widget.TextView;

public class MainActivity extends Activity implements SensorEventListener{

    SensorManager sm;
    Sensor sensor;
    TextView tv;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        tv = (TextView)findViewById(R.id.tv);
        //Get a reference to the sensor manager
        sm = (SensorManager) getSystemService(SENSOR_SERVICE);


        //get reference to the magnetic field sensor
        sensor = sm.getDefaultSensor(Sensor.TYPE_MAGNETIC_FIELD);

        //register a listener
        sm.registerListener(this,sensor,SensorManager.SENSOR_DELAY_NORMAL);

    }
```



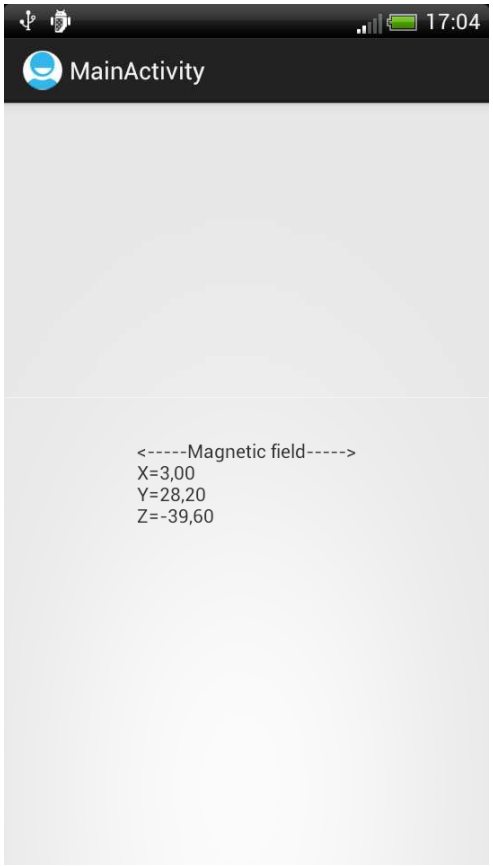
Implements Listener



Sampling rate…

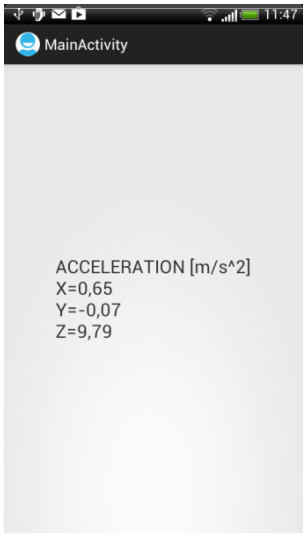
Example

```
public void onAccuracyChanged(Sensor arg0, int arg1) {
    // TODO Auto-generated method stub

}
public void onSensorChanged(SensorEvent event) {
    int type = event.sensor.getType();

    if (type==Sensor.TYPE_MAGNETIC_FIELD){

        float [] M = event.values.clone();

        String msg="<-----Magnetic field----->\n";
        msg+=String.format("X=%.02f\n",M[0])+
        String.format("Y=%.02f\n",M[1])+
        String.format("Z=%.02f",M[2]);

        tv.setText(msg);



    }
```



Example: reading acceleration

```
public class MainActivity extends Activity implements SensorEventListener{

    private SensorManager sm = null;
    private final List<Sensor> deviceSensors = null;
    private TextView tv=null;
    private  Sensor sensor= null;
  @Override
  public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      tv = (TextView)findViewById(R.id.log);
      sm = (SensorManager) getSystemService(SENSOR_SERVICE);
      //deviceSensors = sm.getSensorList(Sensor.TYPE_ALL);
      //Iterator<Sensor> it = deviceSensors.iterator();
      //while (it.hasNext()) tv.setText(tv.getText()+it.next().getName()+"\n");

      sensor = sm.getDefaultSensor(Sensor.TYPE_MAGNETIC_FIELD);
      sm.registerListener(this,sensor,SensorManager.SENSOR_DELAY_NORMAL);
      sensor = sm.getDefaultSensor(Sensor.TYPE_GRAVITY);
      sm.registerListener(this,sensor,SensorManager.SENSOR_DELAY_NORMAL);

  }
  public void onAccuracyChanged(Sensor arg0, int arg1) {

  }
  public void onSensorChanged(SensorEvent event) {
      int type = event.sensor.getType();
      float[] values = event.values.clone();

      if (type == Sensor.TYPE_GRAVITY){
          tv.setTextSize(22);
          String msg= "ACCELERATION [m/s^2]\n"+
                      String.format("X=%.02f\n", values[0])+
                      String.format("Y=%.02f\n", values[1])+
                      String.format("Z=%.02f", values[2]);
          tv.setText(msg);
          return;
      }
```



Example: reading the position

```
package com.example.gpsdemo;

import android.app.Activity;..

public class MainActivity extends Activity {

    TextView tv ;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        tv = (TextView) findViewById(R.id.tv);
        LocationManager lm = (LocationManager) this.getSystemService(Context.LOCATION_SERVICE);
        Criteria criteria = new Criteria();//create an object with no requirements
        String locationProvider = lm.getBestProvider(criteria, false);
        Location location = lm.getLastKnownLocation(locationProvider);

        if (location==null) tv.setText("null!!!!");
        if (location!=null) {
            tv.setText(tv.getText()+"\n"+locationProvider);
            tv.setText(tv.getText()+"\n"+String.valueOf(location.getLatitude()));
            tv.setText(tv.getText()+"\n"+String.valueOf(location.getLongitude()));
        }
    }
}
```