ANDROID APPS DEVELOPMENT FOR MOBILE GAME

Lecture 5: Sensor and Location

Peter Lo

Sensor Overview

- Most Android-powered devices have built-in sensors that measure motion, orientation, and various environmental conditions.
- These sensors are capable of providing raw data with high precision and accuracy, and are useful if you want to monitor three-dimensional device movement or positioning, or you want to monitor changes in the ambient environment near a device.

Categories of Sensors

- Android platform supports three broad categories of sensors:
  - Environmental Sensors
    - These sensors measure various environmental parameters, such as ambient air temperature and pressure, illumination, and humidity (barometers, photometers, and thermometers)
  - Motion Sensors
    - These sensors measure acceleration forces and rotational forces along three axes (Accelerometers, gravity sensors, gyroscopes, and rotational vector sensors)
  - Position Sensors
    - These sensors measure the physical position of a device. (Orientation sensors and magnetometers)

These sensors are hardware-based and are available only if a device manufacturer has built them into a device.

Environment Sensor

- Android provides four sensors that let you monitor various environmental properties.

Sensor Type	Unit	Sensor Event Data	Data Description
TYPE_AMBIENT_TEMPERATURE	°C	SensorEvent.values[0]	Ambient air temperature
TYPE_LIGHT	lx	SensorEvent.values[0]	Illuminance
TYPE_PRESSURE	hPa / mbar	SensorEvent.values[0]	Ambient air pressure
TYPE_RELATIVE_HUMIDITY	%	SensorEvent.values[0]	Ambient relative humidity

Position Sensor

- Android provides the geomagnetic field sensor and the orientation sensor that let you determine the position of a device. It also provide a proximity sensor that lets you determine how close the device to an object.

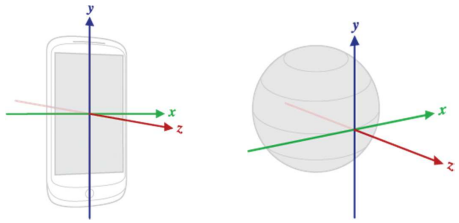
Sensor Type	Unit	Sensor Event Data	Data Description
TYPE_PROXIMITY	cm	SensorEvent.values[0]	Distance from object
TYPE_MAGNETIC_FIELD	μT	SensorEvent.values[0]	Geomagnetic field strength along the x axis
		SensorEvent.values[1]	Geomagnetic field strength along the y axis
		SensorEvent.values[2]	Geomagnetic field strength along the z axis

Motion Sensors Type

Sensor Type	Unit	Sensor Event Data	Data Description
TYPE_ACCELEROMETER	m/s^2	SensorEvent.values[0-2]	Acceleration force along the x, y, z axis
TYPE_GRAVITY	m/s^2	SensorEvent.values[0-2]	Force of gravity along the x, y, z axis.
TYPE_GYROSCOPE	rad/s	SensorEvent.values[0-2]	Rate of rotation around the x, y, z axis.
TYPE_LINEAR_ACCELERATION	m/s^2	SensorEvent.values[0-2]	Acceleration force along the x, y, z axis (excluding gravity).
TYPE_ROTATION_VECTOR		SensorEvent.values[0-3]	Rotation vector component along the x, y, z axis and rotation vector.

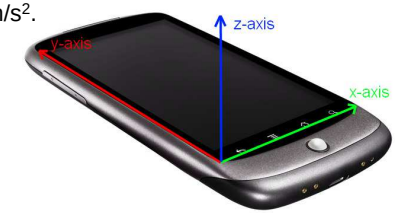
Motion Sensor

- Android provides several sensors that let you monitor the motion of a device.
- Accelerometer and gyroscope sensors are always hardware-based.
- Gravity, linear acceleration and rotation vector sensors can be either hardware-based or software-based .



Acceleration Sensor

- An acceleration sensor measures the acceleration applied to the device, including the force of gravity.
- The force of gravity is always influencing the measured acceleration according to the following relationship:
  - $A = - g - \sum F / mass$
- To measure the real acceleration of the device, the force of gravity must be removed from the accelerometer data because:
  - When the device is sitting on a table, $g = 9.81\ m/s^2$.
  - When the device is in free fall and therefore rapidly accelerating toward the ground at $9.81\ m/s^2$, $g = 0\ m/s^2$.



Sensor Rate

- The rate sensor events are delivered at. This is only a hint to the system. Events may be received faster or slower than the specified rate. Usually events are received faster.
  - SENSOR_DEPLAY_FASTEST – Get sensor data as fast as possible
  - SENSOR_DEPLAY_GAME – Rate suitable for games
  - SENSOR_DEPLAY_NORMAL – Default rate suitable for screen orientation changes
  - SENSOR_DEPLAY_UI – Rate suitable for the user interface (Slowest)

Using Sensor

```
public class MainActivity extends Activity implements SensorEventListener {
    private SensorManager mSensorManager;
    private Sensor mPressure;

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

        mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
        mPressure = mSensorManager.getDefaultSensor(Sensor.TYPE_PRESSURE);
    }

    @Override
    public final void onAccuracyChanged(Sensor sensor, int accuracy) {
        // Do something here if sensor accuracy changes.
    }

    @Override
    public final void onSensorChanged(SensorEvent event) {
        float millibars_of_pressure = event.values[0];
    }

    @Override
    protected void onResume() {
        super.onResume();
        mSensorManager.registerListener(this, mPressure, SensorManager.SENSOR_DELAY_NORMAL);
    }

    @Override
    protected void onPause() {
        super.onPause();
        mSensorManager.unregisterListener(this);
    }
}
```

Used for receiving notifications from the SensorManager when sensor values have changed

Get an instance of the sensor service, and use that to get an instance of a particular sensor

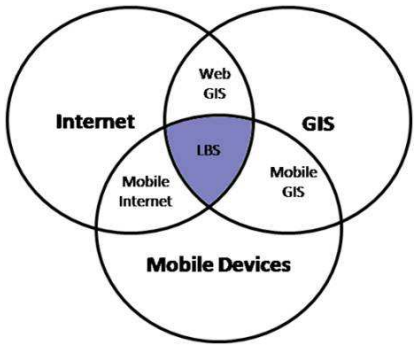
Handling incoming sensor data in the onSensorChanged() callback method

Register a listener for the sensor with specified rate

Unregister the sensor when the activity pauses

Introduction to LBS

- **Location Based Service (LBS)** is an information system driven by the ability of the central system to detect the geographical position of the mobile devices.

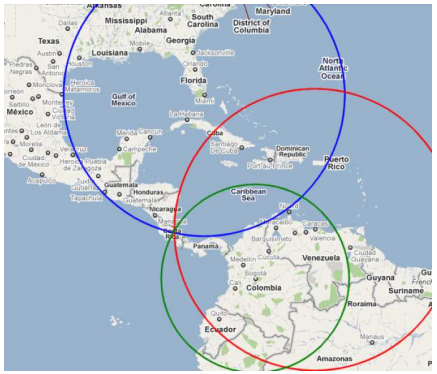


Examples:

- Locate the nearest bank, restaurant, gas station, hotel, golf course, hospital, police station, etc.
- Provide transportation information on how to go from 'here' to 'there'.
- Social networking is used to locate and reach events, friends and family members.

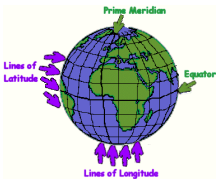
How the GPS Works?

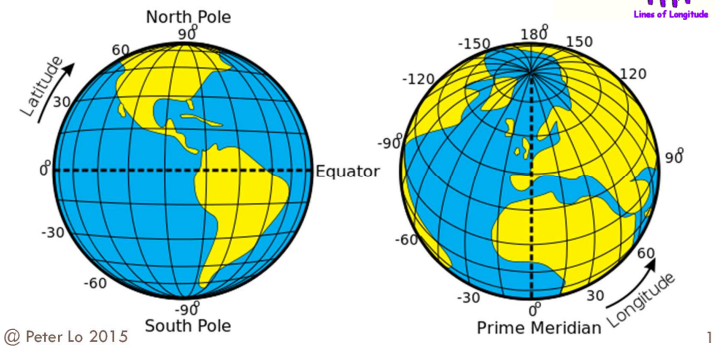
- Created by DOD-USA under the name NAVSTAR (Navigation System for Timing and Ranging) but it is commonly known as **Global Positioning System** (GPS).
- The system's backbone consists of 27 Earth-orbiting satellites (24 in operation and 3 in stand-by mode)
- The three circles intersect on the point over Central America.
- The actual location is: San Jose, Costa Rica.



Latitude and Longitude

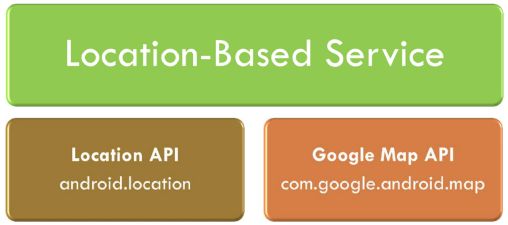
- Latitude in GPS-Decimal notation:
  - +90.00000 (North) to -90.000000 (South)
- Longitude GPS-Decimal notation:
  - +180.000000 (East) to -180.000000 (West)



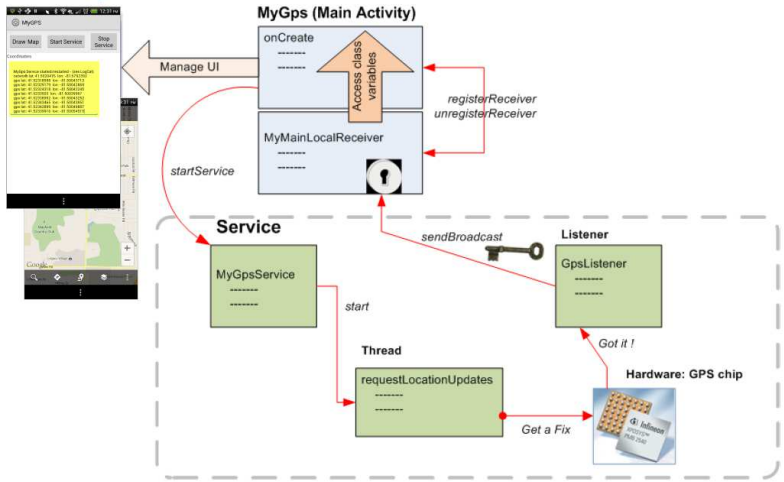


Main Component of LBS

- The Location-Based API includes two packages
  - Google Map API (*com.google.android.maps*)
  - Location API (*android.location*)
- They provide an initial look at the support in the Android platform for building location-based services.
- These API work over the internet to invoke services from Google servers



Android GPS Algorithm



Define Location Listener and Request Location Updates

Define the criteria how to select the location provider

Acquire a reference to the system Location Manager

```
locationManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
criteria = new Criteria();
provider = locationManager.getBestProvider(criteria, false);
LocationListener locationListener = new LocationListener() {
    public void onLocationChanged(Location location) {
        // Make use of new location
    }
    public void onStatusChanged(String provider, int status, Bundle extras) {}
    public void onProviderEnabled(String provider) {}
    public void onProviderDisabled(String provider) {}
};
locationManager.requestLocationUpdates(provider, 0, 0, locationListener);
```

Called when a new location is found by the location provider

Define a listener that responds to location updates

Register the listener with the Location Manager to receive location updates

Requesting User Permissions

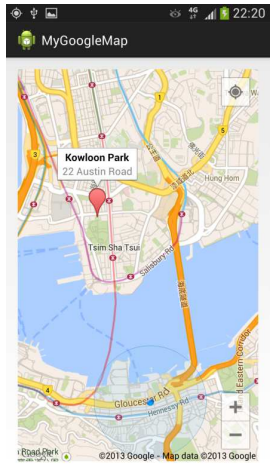
- In order to receive location updates from network provider or GPS provider, you must request user permission by declaring corresponding permission in your Android manifest file.

Permission	Provider
ACCESS_FINE_LOCATION	Allows the API to use the GPS to determine the device's location to within a very small area: • NETWORK_PROVIDER • GPS_PROVIDER
ACCESS_COARSE_LOCATION	Allows the API to use WiFi or mobile cell data (or both) to determine the device's location: • NETWORK_PROVIDER

Without these permissions, your application will fail at runtime when requesting location updates.

Google Maps

- The Google Maps API for Android provides developers with the means to create apps with localization functionality.
- Google Maps API V2 was released at the end of 2012 and it introduced a range of new features including 3D, improved caching, and vector tiles.



Google Maps Android API Getting Started

- Creating a new Android application that uses the Google Maps Android API v2 requires several steps.
- The overall process of adding a map to an Android application is as follows:
  - Download and configure the Google Play services SDK, which includes the Google Maps Android API.
  - Obtain an API key. You will need to register a project in the Google APIs Console, and get a signing certificate for your app.
  - Add the required settings in your application's manifest.
  - Add a map to your application.
  - Publish your application.

If you use the Google Maps Mobile SDK for Business you must download and configure the Google Maps Mobile SDK for Business static library.