Application Development

for the

Android Mobile Platform

Session 3: Advanced APIs

Stanislav Rost
CSAIL, MIT

The Plan

1. Location, Google Maps, Sensor APIs
2. Project Development Session

GPS and Wi-Fi -assisted

LOCATION API

LocationManager

- An Android system service that provides
  - Estimates of current location
  - Alerts of proximity to a specific point
- Getting a LocationManager instance

```
LocationManager locManager = Context.
  getSystemService(Context.LOCATION_SERVICE);
```

- Do not forget to add permissions to the application manifest

```
android.permissions.ACCESS_FINE_LOCATION
android.permissions.ACCESS_COARSE_LOCATION
```

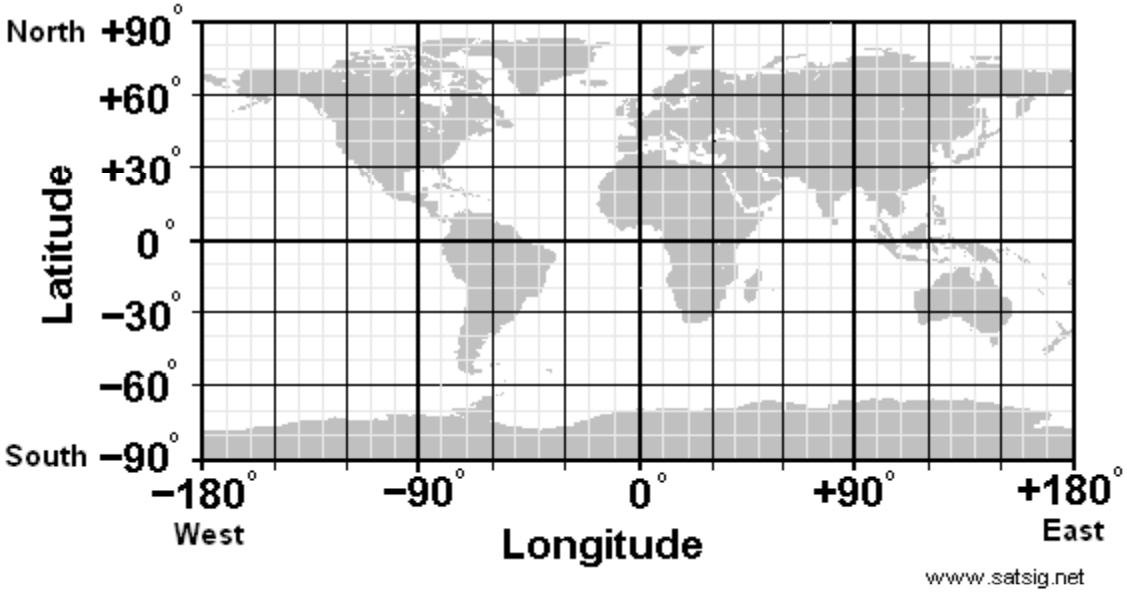
LocationManager Notes

- Can query last known location directly

```
Location here = locManager.getLastKnownLocation("gps");
double lat = here.getLatitude();
double lon = here.getLongitude();
```

- Location and movement work in emulator
  - Load *.KML files using DDMS
- Can supply your own location providers for testing
  - I made a provider that simulates walking to random points

Quick Notes on Cartography



- **Earth is approximately elliptical**
  - One degree is a huge distance
  - Distances to meters: use `Location.distanceTo(Location)`

Location Updates

- Registering for location updates

```
locManager.requestLocationUpdates(String provider,
  long minTime, float minDistance, LocationListener l)
```

- LocationListener callbacks

```
abstract void onLocationChanged(Location loc)

abstract void onProviderDisabled(String provider)

abstract void onProviderEnabled(String provider)

onStatusChanged(String provider, int status, Bundle info)
  status:  OUT_OF_SERVICE
           TEMPORARILY_UNAVAILABLE
           AVAILABLE
```

Proximity Alerts

Register a proximity alert

```
locManager.addProximityAlert(double lat, double lon,
  float radius, long expiration, PendingIntent intent)
```

Designed for a very specific use case

- Intent is meant to start an activity
- Would be more versatile to use `ProximityAlarmListener`

Practical Issues with Location

High initialization, convergence time of GPS

- Up to 10 minutes in Android **1.1** (!!!)
- Activity.onPause() may lose GPS fix
- Android **1.5** introduces improvements using Google's own WiFi location service

Occasionally, very low accuracy (200-500m off)

- Not exclusive to Android hardware
- But, Android hardware has an even smaller antenna

Using the location API in the context of the real world

GOOGLE MAPS API

Getting Started with Google Maps

Google Maps SDK library

- Right-click on project **> Properties > Android >** check off **Google APIs**

Documentation is in a different directory

- **<SDK>/add-ons/google_apis-3/docs/reference/**

Need to import extra packages

- **com.google.android.maps.***

Add the following to the manifest

- <uses-library android:name="com.google.android.maps" />

Google Maps Actors

- **MapActivity**
  - Only this type of **`Activity`** can contain a **`MapView`** in layout
  - Sets up, tears down services for prefetching map images
- **MapView**
  - Renders a Google Map
  - Reacts to user input, zooms and pans the map
- **MapController**
  - Can change the central location and zoom level of a **`MapView`**
- **Overlay**
  - Can draw on a **`MapView`**'s canvas, get access to **`MapView`**

Example: Using GPS with MapView

```
Location myLocation =
   locManager.getLastKnownLocation("gps");

Double lat = myLocation.getLatitude() * 1E6;
Double lon = myLocation.getLongitude() * 1E6;

GeoPoint point =
   new GeoPoint(lat.intValue(), lon.intValue());

//move the map to my location
mapController = mapView.getController();
mapController.animateTo(point);
// if animation is not necessary, this could have been
// mapController.setCenter(point);
```

Wrapping Up MapViews

- Zooming is a bit more complicated

```
mapController.zoomToSpan(int latSpanE6, int lonSpanE6);
```

- Released applications must use developer's Google Maps API key

- Try this tutorial to get started:

  http://developer.android.com/guide/tutorials/views/hello-mapview.html

- If working with maps, look into

  android.location.Geocoder

Interfacing with the real world

SENSOR API

Sensor Framework

SensorManager

```
mSensorManager = (SensorManager)
  Context.getSystemService(Context.SENSOR_SERVICE);

List<Sensor> sensors =
  mSensorManager.getSensorList(Sensor.TYPE_ORIENTATION);

Sensor sensor = sensors.get(0);

mSensorManager.registerListener(sensorEventListener,
          sensor, SensorManager.SENSOR_DELAY_FASTEST);

// SENSOR_DELAY_GAME SENSOR_DELAY_NORMAL
```

Receiving Sensor Events

SensorEventListener

```
abstract void onAccuracyChanged(Sensors, int accuracy)
```

```
SensorManager.SENSOR_ACCURACY_STATUS_[LOW, MEDIUM, HIGH]
```

```
abstract void onSensorChanged(SensorEvent sensorEvent)
```

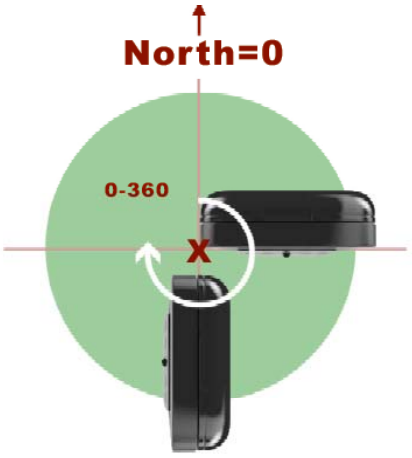
```
int accuracy
Sensor sensor
long timestamp
float[] values
```

For TYPE_ORIENTATION:

```
values[0] : AZIMUTH
values[1]:  PITCH
values[2]:  ROLL
```

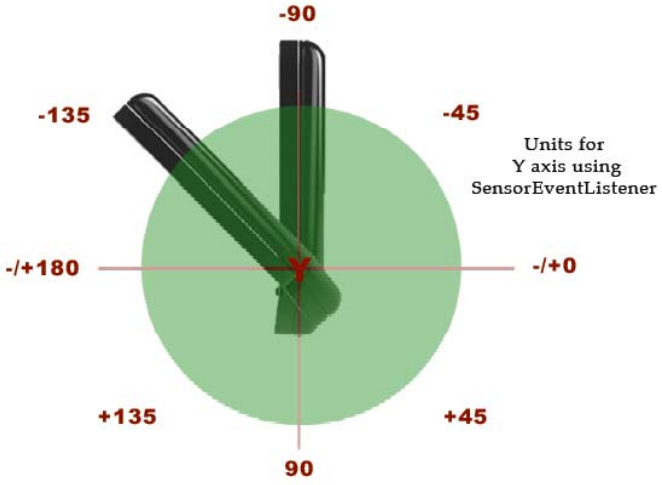
Focus on Orientation Sensors

Azimuth



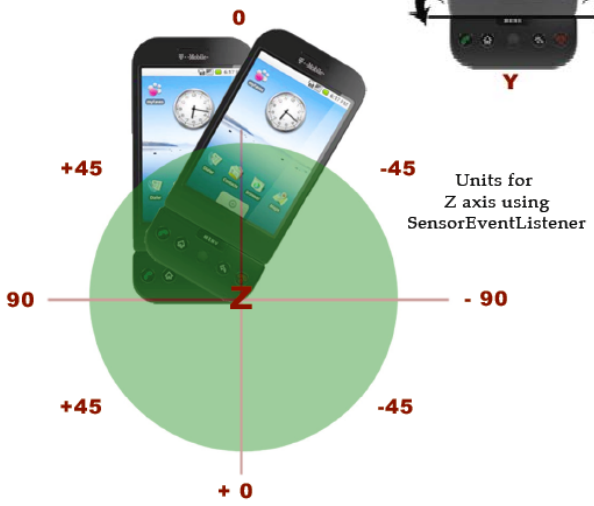
Units for X axis
with SensorEventListener

Pitch



Units for
Y axis using
SensorEventListener

Roll



Units for
Z axis using
SensorEventListener

- Source: novoda.com

Project ideas and their supervised development

HANDS-ON PROGRAMMING

Choose Your Development Track

FingerPainter

- Custom views and touch event processing

Tilt Map

- A MapView that scrolls based on orientation sensors

DistanceWalker

- Uses GPS location to measure and display total walking distance

RoboChat

- HTTP networking and background processing

Finger Painter

- **Level I**
  - Custom View
  - White background
  - Adjust brightness by moving finger up, down
- **Level II**
  - Paint on the screen canvas with your finger
  - Add a palette of colors
  - Make your app survive lifecycle events
- **Level III**
  - Add a Menu option to save image to/load from a PNG file on the SD card

TiltMap

Level I

- A full-screen MapView-based application
- Scroll/zoom the MapView with finger motions

Level II

- Make the application run on a development phone
- Make the application survive the lifecycle events

Level III

- Read orientation sensors to scroll the map by tilting the phone

DistanceWalker

Level I

- Write an application that sums up distance the phone travels between a start-point and an end-point

Level II

- Create an UI that allows users to push buttons for starting and ending a journey
- Display the distance traveled in the UI

Level III

- Draw the travel path on the Google Map

-or-

- Make the distance counter run in the background as a Service

RoboChat

Level I

- Implement the RoboChat UI layout (see Session II slides)

Level II

- Allow users to POST new messages
- Refresh messages from the server

Level III

- Make the app survive the lifecycle events

Summary of Session III

- **Slides are available for download here:**

  `http://www.mit.edu/~stanrost/quanta09/session3/`

- **Topics covered**
  - GPS and Location
  - Google Maps
  - Sensor API

Thank You !

...and good luck in your endeavors!