SENSORS AND LOCATION

ANDROID SENSORS AND LOCATIONS

Sensors

- Available sensors
  - Accelerometer
  - Gyroscope
  - Magnetometer
  - Proximity
  - Temperature
  - Light
  - Microphone
  - Camera
  - GPS

SensorManager

- Most sensors interfaced through SensorManager or LocationManager
  - Obtain pointer to android service using

    ```
    Context.getSystemService( name )
    ```

    - For name, use constant defined by Context class
      - SENSOR_SERVICE for SensorManager
      - LOCATION_SERVICE for LocationManager
- Check for available sensors using

  ```
  List<Sensor> getSensorList(int type)
  ```

  - Type constants provided in Sensor class documentation

SensorManager

- Use getDefaultSensor(int type) to get a pointer to the default sensor for a particular type

```
Sensor accel = sensorManager.getDefaultSensor
                        ( Sensor.TYPE_ACCELEROMETER );
```

- Register for updates of sensor values using

```
registerListener (SensorEventListener, Sensor, rate)
```

  - rate is an int, using one of the following 4 constants
    - SENSOR_DELAY_NORMAL
    - SENSOR_DELAY_UI
    - SENSOR_DELAY_GAME
    - SENSOR_DELAY_FASTEST
  - Use the lowest rate necessary to reduce power usage

SensorManager

- Unregister for sensor events using

```
unregisterListener (SensorEventListener, Sensor)
        or
unregisterListener (SensorEventListener)
```

- Perform register in OnResume() and unregister in OnPause() to prevent using resources while your activity is not visible
- SensorListener is depracated, use SensorEventListener instead
  - See documentation for Sensor, SensorManager, SensorEvent and SensorEventListener

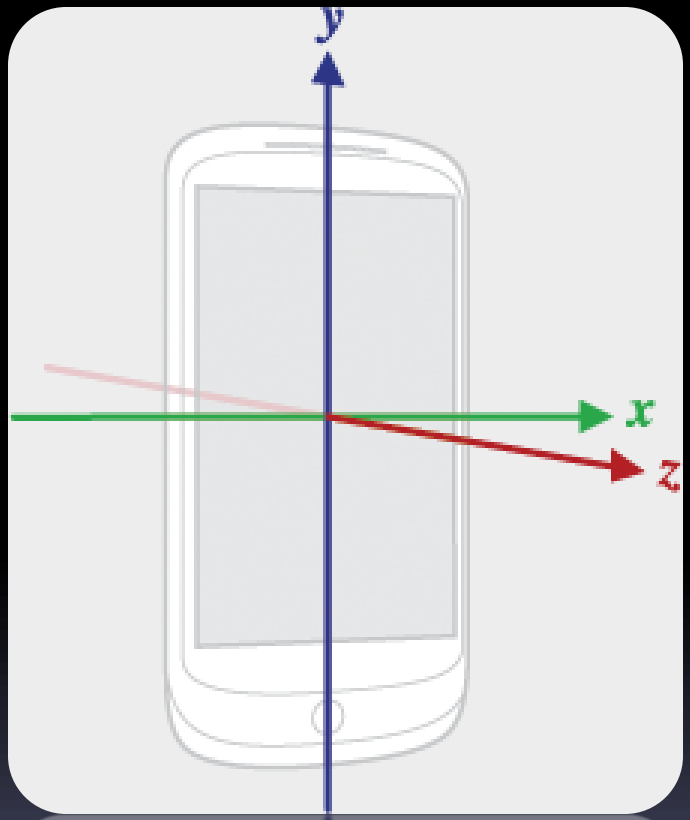
SensorEventListener

- Must implement two methods

```
onAccuracyChanged (Sensor sensor, int accuracy)
onSensorChanged (SensorEvent event)
```

- SensorEvent
  - int accuracy
  - Sensor sensor
  - long timestamp
    - Time in nanoseconds at which event happened
  - float[] values
    - Length and content of values depends on sensor type

Coordinate system



Sensors

- Some sensor types are raw values of physical sensors, others are derived from sensors
- Example raw types
  - TYPE_ACCELEROMETER
  - TYPE_GYROSCOPE
- Example derived values
  - TYPE_LINEAR_ACCELERATION
  - TYPE_ORIENTATION
- Some types depend on the underlying hardware, e.g., TYPE_PROXIMITY may be derived from ambient light sensor

Location

- Providers
  - GPS_PROVIDER
    - GPS - meters
  - NETWORK_PROVIDER
    - Wifi - 100 meters
    - Cellular - 1 km
  - PASSIVE_PROVIDER
    - Piggy-back on other updates
- Each provider has trade-off of power usage, delay, and accuracy
- Use minimum accuracy necessary for application to reduce power usage

LocationManager

- Register for location updates using

  `void requestLocationUpdates (. . .)`

  - Multiple options for request arguments, based on type of request needed. See documentation for all options.
- Typically pass a LocationListener (like SensorEventListener)

LocationListener

- Methods

```
onLocationChanged (Location location)
onProviderDisabled (String provider)
onProviderEnabled (String provider)
onStatusChanged (String Provider, int status,
                                  Bundle extras)
```

- Provider is the technology used for location measurements

Request permissions

- Register permission in manifest file
  - android.permission.ACCESS_FINE_LOCATION
    - If (possibly) using GPS
  - android.permission.ACCESS_COARSE_LOCATION
    - For any other location access

```
<manifest ... >
    <uses-permission android:name=
          "android.permission.ACCESS_FINE_LOCATION" />
    ...
</manifest>
```

Energy usage

- Reduce power usage wherever possible
- Unregister listener when new updates are not needed
  - removeUpdates (LocationListener)
- Reduce frequency of updates
  - Use higher threshold values for minimum distance and time interval
- Restrict providers used
  - Use lower cost providers when possible

MapView

- Interface to Google Maps API
- Maps external library not part of standard Android library
  - Install Google API in SDK and AVD Manager
  - Select Google API as target for project
  - Must register for Google Maps API key

    http://code.google.com/android/add-ons/google-apis/mapkey.html
- MapActivity provides subclass of Activity linked to MapView (similar to ListActivity)

Request permissions

- Add permission and library requests in manifest file
  - Include location permission as before
  - android.permission.INTERNET
    - To interface Google Maps API
  - com.google.android.maps
    - For Google maps library

```
<manifest ... >
   <uses-permission android:name="android.permission.INTERNET"/>
   <application ... >
      <uses-library android:name="com.google.android.maps" />
      ...
   </application>
   ...
</manifest>
```

Map API key

- Add Google Map API key to MapView
  - In layout xml file, add following attribute to MapView item

```
<com.google.android.maps.MapView
        . . .
        android:apiKey="your key here"
/>
```

MapView class

- Enable and configure zoom controls
- Obtain MapController for control of map
- Obtain overlay list to add new overlay
- Set map mode
  - Satellite
  - Street view
  - Traffic

MapController class

- Animate/pan map
- Zoom
  - In
  - Out
  - Set
- Re-center map

GeoPoint class

- A latitiude/longitude position stored as integer numbers in microdegrees
  - `degrees × 1E6`

Overlay class

- Add overlay images or icons at GeoPoints
- Abstract class, use MyLocationOverlay or subclass ItemizedOverlay
- MyLocationOverlay
  - Automatically provide location tracking and indicate position with blue dot
- ItemizedOverlay
  - Subclass to provide a list of overlay icons
    - Can implement touch events

Testing

- Limited simulation support for sensors
- To test Android application on device, add debuggable attribute to manifest file

```
<manifest ... >
   <application ...
      android:debuggable="true" >
      ...
   </application>
   ...
</manifest>
```