CMSC 628/491: Introduction to Mobile Computing
Android Sensors, and Location

Nilanjan Banerjee

University of Maryland
Baltimore County
nilanb@umbc.edu

Android Sensors Overview

- Android Sensors:
- MIC
- Camera
- Temperature
- Location (GPS or Network)
- Orientation
- Accelerometer
- Proximity
- Pressure
- Light

Two types of sensors on the android platform

- Hardware sensors
  - Physical sensors present on the phone
  - Accelerometers, temperature, gyroscope

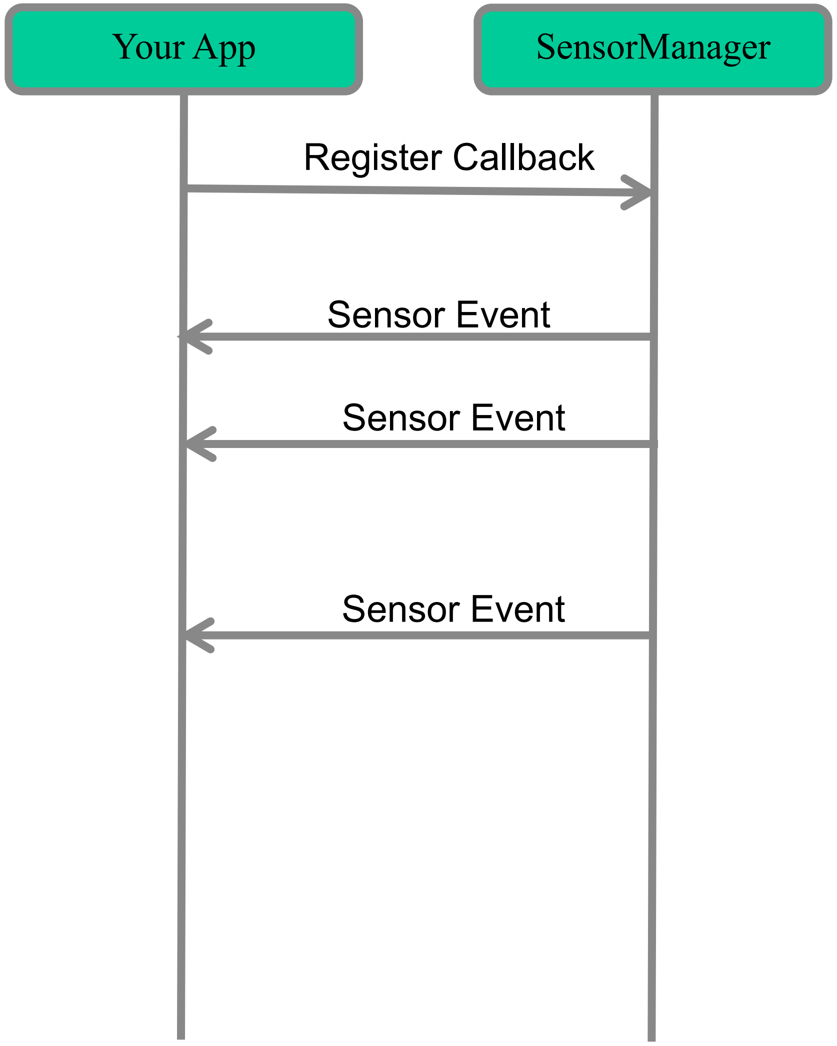
- Software sensors
  - Virtual sensors that are built on top of hardware sensors.
  - Orientation sensors --- accelerometer + gyroscope?

Async Callbacks

Android's sensors are controlled by external services and only send events when they choose to

- An app must register a callback to be notified of a sensor event

- Each sensor has a related XXXListener interface that your callback must implement
  - E.g. LocationListener



Getting the Relevant System Service

- The non-media (e.g. not camera) sensors are managed by a variety of XXXXManager classes:
  - LocationManager (GPS)
  - SensorManager (accelerometer, gyro, proximity, light, temp)
- The first step in registering is to obtain a reference to the relevant manager
- Every Activity has a getSystemService() method that can be used to obtain a reference to the needed manager

```
public class MyActivity … {

  private SensorManager sensorManager_;

  public void onCreate(){
     …

     sensorManager_ = (SensorManager) getSystemService(SENSOR_SERVICE);
  }

}
```

Registering for Sensor Updates

- The SensorManager handles registrations for
  - Accelerometer, Temp, Light, Gyro
- In order for an object to receive updates from a sensor, it must implement the SensorEventListener interface
- Once the SensorManager is obtained, you must obtain a reference to the specific sensor you are interested in updates from
- The arguments passed into the registerListener method determine the sensor that you are connected to and the rate at which it will send you updates

```
public class MyActivity … implements SensorListener{
  private Sensor accelerometer_;
  private SensorManager sensorManager_;

  public void connectToAccelerometer() {
          sensorManager_ = (SensorManager)getSystemService(SENSOR_MANAGER);
          accelerometer_ = sensorManager_
                                        .getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
          sensorManager_.registerListener(this, accelerometer_,
                                        SensorManager.SENSOR_DELAY_NORMAL);


}
```

The SensorEventListener Interface

- Because there is one interface for multiple types of sensors, listening to multiple sensors requires switching on the type of event (or creating separate listener objects)
- Also forces registration at the same rate per listener
- Simple approach:

```
public class MyActivity … implements SensorListener{


          // Called when a registered sensor changes value
          @Override
          public void onSensorChanged(SensorEvent sensorEvent) {
                    if (sensorEvent.sensor.getType() == Sensor.TYPE_ACCELEROMETER) {
                              float xaccel = sensorEvent.values[0];
                              float yaccel = sensorEvent.values[1];
                              float zaccel = sensorEvent.values[2];


                    }
          }


          // Called when a registered sensor's accuracy changes
          @Override
          public void onAccuracyChanged(Sensor arg0, int arg1) {
                    // TODO Auto-generated method stub

          }
}
```

The SensorEventListener Interface

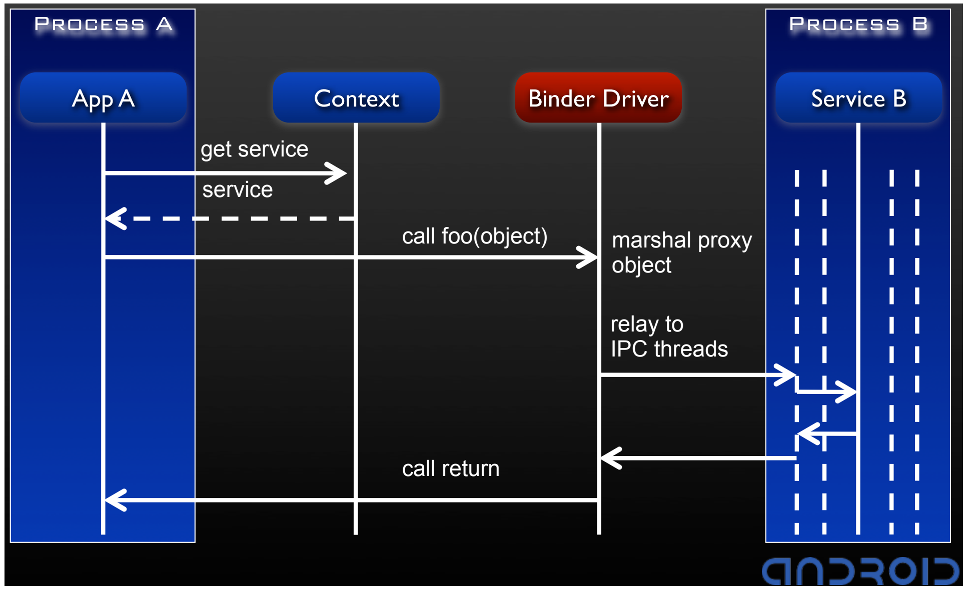
- Another approach for multiple sensors (probably better):

```
public class MyActivity … {

         private class AccelListener implements SensorListener {
                    public void onSensorChanged(SensorEvent sensorEvent) {
                              …
                    }
                    public void onAccuracyChanged(Sensor arg0, int arg1) {}
          }

          private class LightListener implements SensorListener {
                    public void onSensorChanged(SensorEvent sensorEvent) {
                              …
                    }
                    public void onAccuracyChanged(Sensor arg0, int arg1) {}
          }

          private SensorListener accelListener_ = new AccelListener();
          private SensorListener lightListener_ = new LightListener();

          …
          public void onResume(){
             …
           sensorManager_.registerListener(accelListener, accelerometer,
                                              SensorManager.SENSOR_DELAY_GAME);
           sensorManager_.registerListener(lightListener, lightsensor,
                                              SensorManager.SENSOR_DELAY_NORMAL);

          }
          public void onPause(){
            sensorManager_.unregisterListener(accelListener_);
            sensorManager_.unregisterListener(lightListener_);
          }
```

Android System Services

- Each App runs in its own process
- Each Android system service, such as the SensorManager, also runs in its own thread

- This has important implications:
  1. Communication with the system services is through IPC
  2. The thread that delivers an event will be a special thread that is dedicated to processing incoming IPC calls
  3. If you directly update the GUI from any thread other than the display thread, bad things happen



How to Update the GUI with Sensor Data

- Android has a built in mechanism for queuing work that needs to be run on the display thread
- The Handler class allows you to create a queue inside of your Activity that can store work for the display thread
- You create a handler once and then post work to it

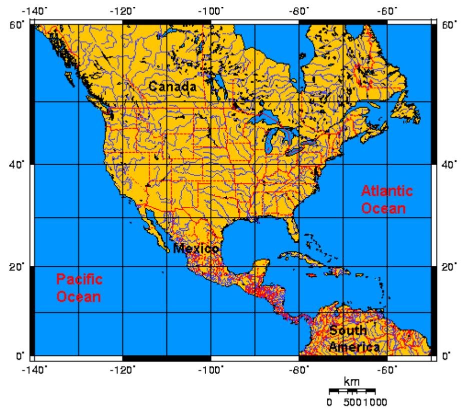
```
public class MyActivity … implements SensorListener{
          private class AccelWork implements Runnable {
               private Location data_;
               public AccelWork(Location d){data_ = d;}

               public void run(){
                    //do something with the data to the GUI
               }
          }

          private Handler myHandler_ = new Handler();

          // Called when a registered sensor changes value
          @Override
          public void onSensorChanged(SensorEvent sensorEvent) {
               AccelWork work = new AccelWork(sensorEvent);
               myHandler_.post(work);
          }
}
```

How to Update the GUI with Sensor Data

- The lazy approach

```
public class MyActivity … implements SensorListener{

          private Handler myHandler_ = new Handler();

          // Called when a registered sensor changes value
          @Override
          public void onSensorChanged(final SensorEvent sensorEvent) {
              myHandler_.post(
                 new Runnable(){
                    public void run(){
                       //do something with the sensorEvent data
                       //to the gui
                    }
                 }
              );
          }
}
```

What is localization, aka location?



absolute location
(lat, long)

"I hope this bullhorn will make this meeting a little less boring."

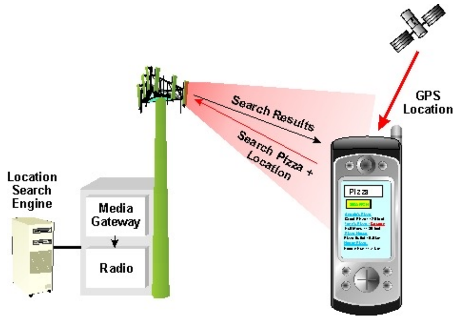
relative location

context location

Why should I care about localization?

Rummble for iPhone
Download it here
...free from the App store

social application



advertisements

podcasting

Registering for Location Updates

- The LocationManager handles registrations for GPS and network location updates
- In order for an object to receive updates from GPS, it must implement the LocationListener interface
- Once the LocationManager is obtained, an object registers for updates by calling requestLocationUpdates (there are multiple versions you can use)
- The arguments passed into the requestLocationUpdates method determine the granularity of location changes that will generate an event
  - send updates that are at least X meters apart
  - send updates at least this far apart in time
  - send updates that have this minimum accuracy

```
public class MyActivity … implements LocationListener{

  private LocationManager locationManager_;

  public void onCreate(){
     …

     locationManager_ = (LocationManager) getSystemService(LOCATION_SERVICE);
     locationManager_.requestLocationUpdates(LocationManager.GPS_PROVIDER, 10,
                                             Criteria.ACCURACY_FINE, this);
  }

}
```

Location Providers

- The phone’s location can be determined from multiple providers
  - GPS
  - Network
- GPS location updates consume significantly more power than network location updates but are more accurate
  - GPS: 25 seconds * 140mA = 1mAh
  - Network: 2 seconds * 180mA = 0.1mAh
- The provider argument determines which method will be used to get a location for you
- You can also register for the PASSIVE_PROVIDER which only updates you if another app is actively using GPS / Network location

```
public class MyActivity … implements LocationListener{

  private LocationManager locationManager_;

  public void onCreate(){
     …

     locationManager_ = (LocationManager) getSystemService(LOCATION_SERVICE);
     locationManager_.requestLocationUpdates(LocationManager.PASSIVE_PROVIDER, 10,
                                            Criteria.ACCURACY_FINE, this);
  }

}
```

The LocationListener Interface

```
public class MyActivity … implements LocationListener{

   …
              // Called when your GPS location changes
              @Override
              public void onLocationChanged(Location location) {

              }

              // Called when a provider gets turned off by the user in the settings
              @Override
              public void onProviderDisabled(String provider) {


              }

              // Called when a provider is turned on by the user in the settings
              @Override
              public void onProviderEnabled(String provider) {


              }

              // Signals a state change in the GPS (e.g. you head through a tunnel and
              // it loses its fix on your position)
              @Override
              public void onStatusChanged(String provider, int status, Bundle extras) {


              }
}
```

Location Information

```
public class MyActivity … implements LocationListener{

   …
            // Called when your GPS location changes
            @Override
            public void onLocationChanged(Location location) {

                        double altitude = location.getAltitude();
                        double longitude = location.getLongitude();
                        double latitude = location.getLatitude();
                        float speed = location.getSpeed();
                        float bearing = location.getBearing();
                        float accuracy = location.getAccuracy(); //in meters
                        long time = location.getTime(); //when the fix was obtained

                        // Other useful Location functions:
                        //
                        // location.distanceTo(dest)
                        // location.bearingTo(dest)
            }


}
```

Being a Good Citizen…

- It is very important that you unregister your App when you no longer need updates
- For example, you should always unregister your listener when your Activity is paused!
- If you unregister when you pause, you must also reregister when you resume
  - This is true for all sensors!

```
public class MyActivity … {

          private LocationManager locationManager_;

          public void onCreate(Bundle savedInstanceState) {
                    …
                    locationManager_ = (LocationManager)getSystemService(LOCATION_SERVICE);
          }
          protected void onPause() {
                    super.onPause();
                    locationManager_.removeUpdates(this);
          }
          protected void onResume() {
                    super.onResume();
                    locationManager_.requestLocationUpdates(LocationManager.GPS_PROVIDER, 10,
                                                            Criteria.ACCURACY_FINE, this);
          }
…
}
```

CMSC 628/491: Introduction to Mobile Computing
Android Sensors, and Location

Nilanjan Banerjee

University of Maryland
Baltimore County
nilanb@umbc.edu