Mobile Application (Design and) Development

11th class

Prof. Stephen Intille
s.intille@neu.edu

Q&A

Today

- Services
- Location and sensing
- Design paper presentations
  - Tapworthy Chapter 2
    Presenter: Naveen Babu
  - Tapworthy Chapter 3
    Presenter: Alex Nikolayev

Schedule

- **Today: design assignment 3 due**
  - I will review this weekend and provide some feedback Monday at the latest
- **Sunday: Programming assignment 3 due**
- **Wednesday:**
  - Design assignment 4: In-Class Presentation (team)
  - Pecha Kucha format. Example for Project Presentations on website

Last time: services

- Required for using location and other sensors

Location service

- Location Manager
  - Obtain current location
  - Track movement
  - Set proximity alerts for areas
  - Find available Location Providers
- Location Providers
  - Various location-finding technologies

Emulator challenges

- Emulator has limited sensing simulation capabilities
  - Location Controls in DDMS in Eclipse
  - OpenIntents Sensor Simulator
- Best to use/borrow a real device

Different phones/options

- Android can pick an option

```
Criteria criteria = new Criteria();
criteria.setAccuracy(Criteria.ACCURACY_COARSE);
criteria.setPowerRequirement(Criteria.POWER_LOW);
criteria.setAltitudeRequired(false);
criteria.setBearingRequired(false);
criteria.setSpeedRequired(false);
criteria.setCostAllowed(true);
```

LocationListener

```
String provider = LocationManager.GPS_PROVIDER;

int t = 5000; // milliseconds
int distance = 5; // meters

LocationListener myLocationListener = new LocationListener() {

  public void onLocationChanged(Location location) {
    // Update application based on new location.
  }

  public void onProviderDisabled(String provider){
    // Update application if provider disabled.
  }

  public void onProviderEnabled(String provider){
    // Update application if provider enabled.
  }

  public void onStatusChanged(String provider, int status,  Bundle extras){
    // Update application if provider hardware status changed.
  }
};

locationManager.requestLocationUpdates(provider, t, distance, myLocationListener);
```

Proximity alert BR

```
public class ProximityIntentReceiver extends BroadcastReceiver {

  @Override
  public void onReceive (Context context, Intent intent) {
    String key = LocationManager.KEY_PROXIMITY_ENTERING;

    Boolean entering = intent.getBooleanExtra(key, false);
    [ ... perform proximity alert actions ... ]
  }

}

privte void register() {
  IntentFilter filter = new IntentFilter(TREASURE_PROXIMITY_ALERT);
  registerReceiver(new ProximityIntentReceiver(), filter);
}
```

Reverse Geocoding

```
location = locationManager.getLastKnownLocation(LocationManager.GPS_PROVIDER);

double latitude = location.getLatitude();
double longitude = location.getLongitude();
List<Address> addresses = null;

Geocoder gc = new Geocoder(this, Locale.getDefault());
try {
  addresses = gc.getFromLocation(latitude, longitude, 10);
} catch (IOException e) {}
```

Geocoding an address

```
Geocoder fwdGeocoder = new Geocoder(this, Locale.US);
String streetAddress = "160 Riverside Drive, New York, New York";

List<Address> locations = null;
try {
  locations = fwdGeocoder.getFromLocationName(streetAddress, 10);
} catch (IOException e) {}
```

Sensing

- Device specific
- ServiceManager provides access to Sensor Manager Service

```
String service_name = Context.SENSOR_SERVICE;
SensorManager sensorManager = (SensorManager) getSystemService(service_name);
```

Sensors

- Sensor.TYPE_ACCELEROMETER
  - Acceleration 3-axes m/s^2
- Sensor.TYPE_GYROSCOPE
  - 3 axis orientation in degrees
- Sensor.TYPE_LIGHT
  - Single value in lux
- Sensor.TYPE_MAGNETIC_FIELD
  - Microteslas in 3-axes

Sensors

- Sensor.TYPE_ORIENTATION
  - Orientation in 3-axes in degrees
- Sensor.TYPE_PRESSURE
  - Single value in kilopascals
- Sensor.TYPE_PROXIMITY
  - Distance in meters
- Sensor.TYPE_TEMPERATURE
  - Value degrees Celsius

Values (and more info)

http://developer.android.com/reference/android/hardware/Sensor.html

A bit about accelerometer data

- Lateral (x)
- Longitudinal (y)
- Vertical (z) (out of screen)

Checking for sensors

Sensor defaultGyroscope = sensorManager.getDefaultSensor (Sensor.TYPE_GYROSCOPE);

(Returns null if none)

Or, get a list of all sensors of a type:

List<Sensor> pressureSensors = sensorManager.getSensorList(Sensor.TYPE_PRESSURE);

Or, get a list of all sensors of a type:

List<Sensor> allSensors = sensorManager.getSensorList(Sensor.TYPE_ALL);

Listening for sensors

```
final SensorEventListener mySensorEventListener = new SensorEventListener() {
  public void onSensorChanged(SensorEvent sensorEvent) {
    // TODO Monitor Sensor changes.
  }

  public void onAccuracyChanged(Sensor sensor, int accuracy) {
    // TODO React to a change in Sensor accuracy.
  }
};
```

- Accuracy:
  - SensorManager.SENSOR_STATUS_ACCURACY_LOW
  - SensorManager.SENSOR_STATUS_ACCURACY_MEDIUM
  - SensorManager.SENSOR_STATUS_ACCURACY_HIGH
  - SensorManager.SENSOR_STATUS_ACCURACY_UNRELIABLE

Checking for sensors

```
public void setupSensorListener() {
  SensorManager sm = (SensorManager)getSystemService(Context.SENSOR_SERVICE);
  int sensorType = Sensor.TYPE_ACCELEROMETER;
  sm.registerListener(mySensorEventListener,
                      sm.getDefaultSensor(sensorType),
                      SensorManager.SENSOR_DELAY_NORMAL);
}

final SensorEventListener mySensorEventListener = new SensorEventListener() {
  public void onSensorChanged(SensorEvent sensorEvent) {
    if (sensorEvent.sensor.getType() == Sensor.TYPE_ACCELEROMETER) {
      float xAxis_lateralA = sensorEvent.values[0];
      float yAxis_longitudinalA = sensorEvent.values[1];
      float zAxis_verticalA = sensorEvent.values[2];
      // TODO apply the acceleration changes to your application.
    }
  }
};
```

Register

```
// Usually in onResume

Sensor sensor = sensorManager.getDefaultSensor(Sensor.TYPE_PROXIMITY;
sensorManager.registerListener(mySensorEventListener, sensor,
SensorManager.SENSOR_DELAY_NORMAL);


// Usually in onPause

sensorManager.unregisterListener(mySensorEventListener)
```

Listener for changes (accel)

```
public void setupSensorListener() {
  SensorManager sm = (SensorManager)getSystemService(Context.SENSOR_SERVICE);
  int sensorType = Sensor.TYPE_ACCELEROMETER;
  sm.registerListener(mySensorEventListener,
                      sm.getDefaultSensor(sensorType),
                      SensorManager.SENSOR_DELAY_NORMAL);
}

final SensorEventListener mySensorEventListener = new SensorEventListener() {
  public void onSensorChanged(SensorEvent sensorEvent) {
    if (sensorEvent.sensor.getType() == Sensor.TYPE_ACCELEROMETER) {
      float xAxis_lateralA = sensorEvent.values[0];
      float yAxis_longitudinalA = sensorEvent.values[1];
      float zAxis_verticalA = sensorEvent.values[2];
      // TODO apply the acceleration changes to your application.
    }
  }
};
```

Example

Orientation

```
field sensors
float[] accelerometerValues;
float[] magneticFieldValues;

final SensorEventListener myAccelerometerListener = new SensorEventListener() {
  public void onSensorChanged(SensorEvent sensorEvent) {
    if (sensorEvent.sensor.getType() == Sensor.TYPE_ACCELEROMETER)
      accelerometerValues = sensorEvent.values;
  }

  public void onAccuracyChanged(Sensor sensor, int accuracy) {}
};

final SensorEventListener myMagneticFieldListener = new SensorEventListener() {
  public void onSensorChanged(SensorEvent sensorEvent) {
    if (sensorEvent.sensor.getType() == Sensor.TYPE_MAGNETIC_FIELD)
      magneticFieldValues = sensorEvent.values;
  }

  public void onAccuracyChanged(Sensor sensor, int accuracy) {}
};
```

Orientation

```
public void connectListeners() {
  SensorManager sm = (SensorManager)getSystemService(Context.SENSOR_SERVICE);
  Sensor aSensor = sm.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
  Sensor mfSensor = sm.getDefaultSensor(Sensor.TYPE_MAGNETIC_FIELD);

  sm.registerListener(myAccelerometerListener,
                      aSensor,
                      SensorManager.SENSOR_DELAY_UI);

  sm.registerListener(myMagneticFieldListener,
                      mfSensor,
                      SensorManager.SENSOR_DELAY_UI);
}
```

Controlling vibration

```
String vibratorService = Context.VIBRATOR_SERVICE;
Vibrator vibrator = (Vibrator)getSystemService(vibratorService);

long[] pattern = {1000,  2000, 4000, 8000, 16000 };
vibrator.vibrate(pattern, 0); // Execute vibration pattern.
vibrator.vibrate(1000); // Vibrate for 1 second.
```

Easy!

Location service

- Location Manager

```
// Implicitly start a Service
Intent myIntent = new Intent(MyService.ORDER_PIZZA);
myIntent.putExtra("TOPPING", "Margherita");
startService(myIntent);

// Explicitly start a Service
startService(new Intent(this, MyService.class));
```

(To use this example, would need to include a MY_ACTION constant in MyService class and use an Intent Filter to register the Service as a provider of MY_ACTION)