# Android Programing

Gavrilut Dragos

# Sensors

- Hardware components that can provide informations about motion, environment, position and so on.
- Not all of the sensors curently suported by the Android platform are availabe on all devices and not all versions of Android suports the software equavalent of the hardware components.
- A sensor can be either hardware (there is a phisical component that can measure and provide data) or software. A software sensor that gather its data from one or more hardware sensors.

# Sensors

| Sensor | Type | Common Uses |
|---|---|---|
| TYPE_ACCELEROMETER | Hardware | Motion detection (shake, tilt, etc.). |
| TYPE_AMBIENT_TEMPERATURE | Hardware | Monitoring air temperatures. |
| TYPE_GRAVITY | Software or Hardware | Motion detection (shake, tilt, etc.). |
| TYPE_GYROSCOPE | Hardware | Rotation detection (spin, turn, etc.). |
| TYPE_LIGHT | Hardware | Controlling screen brightness. |
| TYPE_LINEAR_ACCELERATION | Software or Hardware | Monitoring acceleration along a single axis. |
| TYPE_MAGNETIC_FIELD | Hardware | Creating a compass. |
| TYPE_ORIENTATION | Software | Determining device position. |
| TYPE_PRESSURE | Hardware | Monitoring air pressure changes. |
| TYPE_PROXIMITY | Hardware | Phone position during a call. |
| TYPE_RELATIVE_HUMIDITY | Hardware | Monitoring dewpoint, absolute, and relative humidity. |
| TYPE_ROTATION_VECTOR | Software or Hardware | Motion detection and rotation detection. |
| TYPE_TEMPERATURE | Hardware | Monitoring temperatures. |

# Sensors

- Objects that are related to sensors:
  - SensorManager
  - Sensor
  - SensorEvent
  - SensorEventListener

# Sensors

- 2 type of sensors:
  - Streamimg sensors (return data at a specific time interval)
  - Non-streaming – it reports data when a change occurs in the its data.
- Use Sensor. getMinDelay() method to find out the type of sensor: 0 means non-streaming

# Sensors

```
public class MainActivity extends Activity implements SensorEventListener {
    private Sensor sensorAccelerometer;
    private SensorManager sensorManager;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        sensorManager = (SensorManager)getSystemService(SENSOR_SERVICE);
        sensorAccelerometer = sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
         sensorManager.registerListener(this, sensorAccelerometer, SensorManager.SENSOR_DELAY_NORMAL);
    }
    protected void onResume() {
        if (sensorManager!=null)
            sensorManager.registerListener(this, sensorAccelerometer, SensorManager.SENSOR_DELAY_NORMAL);
    }
    protected void onPause() {
        if (sensorManager!=null)
            sensorManager.unregisterListener(this);
    }
    @Override
    public void onAccuracyChanged(Sensor s, int value) {
        Log.v("TAG","Accuracy Changed");
        Log.v("TAG","Sensor = "+s.getName());
        Log.v("TAG","Vendor = "+s.getVendor());
        Log.v("TAG","Version = "+String.valueOf(s.getVersion()));
        Log.v("TAG","Value = "+String.valueOf(value));
    }
    @Override
    public void onSensorChanged(SensorEvent s) {
       Log.v("TAG","Sensor Changed");
       Log.v("TAG","Sensor = "+s.sensor.getName());
       String values = "";
       for (int tr=0;tr<s.values.length;tr++)
           values+=String.format("%f,", s.values[tr]);
       Log.v("TAG","Values = "+values);
    }
```

# Sensors

- Result (Samsung Galaxy Tablet 10.1)
  - Accuracy Changed
    - Sensor = MPL accel
    - Vendor = Invensense
    - Version = 1
  - Sensor Changed
    - Sensor = MPL accel
    - Values = 0.000000,-0.775986,9.886641,
- Values field holds the data that is specific for each sensor

# Sensor values

- Sensor.TYPE_ACCELEROMETER
  - 3 values measured in SI units
- Sensor.TYPE_MAGNETIC_FIELD:
  - 3 values measured in micro-Tesla units
- Sensor.TYPE_GYROSCOPE
  - 3 values measured in radians/second
- Sensor.TYPE_LIGHT
  - One unit measured in SI lux unit
- Sensor.TYPE_PRESSURE
  - One unit measuring atmospheric presure in hPA
- Sensor.TYPE_PROXIMITY
  - One unit measured in centimeters
- More details on: http://developer.android.com/reference/android/hardware/SensorEvent.html

# Sensors

```
public class MainActivity extends Activity implements SensorEventListener {
    private SensorManager sensorManager;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        sensorManager = (SensorManager)getSystemService(SENSOR_SERVICE);
        List<Sensor> sensorList = sensorManager.getSensorList(Sensor.TYPE_ALL);
        for (Sensor s: sensorList)
        {
          Log.v("TAG",String.format("Name:%s, Vendor:%s,
                                    Type:%d",s.getName(),s.getVendor(),s.getType()));
        }
    }
}
```

## Samsung Galaxy Tablet 10.1

Name:BH1721FVC Light Sensor,Vendor:ROHM,Type:5
Name:MPL rotation vector,Vendor:Invensense,Type:11
Name:MPL linear accel,Vendor:Invensense,Type:10
Name:MPL gravity,Vendor:Invensense,Type:9
Name:MPL Gyro,Vendor:Invensense,Type:4
Name:MPL accel,Vendor:Invensense,Type:1
Name:AK8975 Magnetic field Sensor,Vendor:Asahi Kasei Microdevices,Type:2
Name:AK8975 Orientation Sensor,Vendor:Asahi Kasei Microdevices,Type:3

# Location

- Location services enables you to find your location based on a GPS or a network.
- Requires the following permisions:
  - *android.permission.ACCESS_FINE_LOCATION*
  - *android.permission.INTERNET*
- *Use: LocationManager* to obtain informations about the current location

# Location

```
public class MainActivity extends Activity implements LocationListener {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        LocationManager lm = (LocationManager) this.getSystemService(Context.LOCATION_SERVICE);
        boolean gps = lm.isProviderEnabled(LocationManager.GPS_PROVIDER);
        boolean net = lm.isProviderEnabled(LocationManager.NETWORK_PROVIDER);
        Log.v("TAG","GPS:"+String.valueOf(gps)+" Net:"+String.valueOf(net));
        lm.requestLocationUpdates(LocationManager.GPS_PROVIDER, 3000, 5, this);
    }
    @Override
    public void onLocationChanged(Location l) {
        Log.v("TAG",String.format("Location changed: Long:%f,
                                   Lat:%f",(float)l.getLongitude(),(float)l.getLatitude()));
    }
    @Override
    public void onProviderDisabled(String txt) {
        Log.v("TAG",String.format("Provider disabled: %s",txt));
    }
    @Override
    public void onProviderEnabled(String txt) {
        Log.v("TAG",String.format("Provider enabled: %s",txt));
    }
    @Override
    public void onStatusChanged(String arg0, int arg1, Bundle arg2) {
        Log.v("TAG","Status Changed");
    }
}
```

# Location

- Sometimes location can be cached. In this cases it is recommended that you have an update system that rejects some old locations (if you receive a location at an time interval that is higher than a specific rate (for example, more than 10 seconds) you can assume that that location may not be correct and require a new location again.

# Sound

- Android system supports different sound types (mp3, mp4, ogg , etc)
- There are two ways an application can use the sound manager of Android:
  - Use the MediaPlayer class (usually for larger sound)
  - Use SoundPool class (for small sound ~ a couple of seconds)

# SoundPool

- Constructor:
  public SoundPool (int maxStreams, int streamType, int srcQuality)
- Load a stream from resources:
  public int load (Context context, int resId, int priority)
- Play a stream
  public final int play (int soundID, float leftVolume, float rightVolume, int priority, int loop, float rate)
- Sound resources can be put in the "res/raw" folder.

# Sound - example

```
SoundPool sp = new SoundPool(1, AudioManager.STREAM_MUSIC, 0);
if (sp!=null)
{
        int streamID = sp.load((Context)this,R.raw.my_sound,1);
        float volume = 0.5f;
        sp.play(streamID,volume,volume,0,0,1);
        …
        sp.stop(streamID);
        sp.unload(streamID);
        sp.release();
        sp = null;
}
```

```
MediaPlayer mp = MediaPlayer.create((Context)this,R.raw.my_sound);
if (mp!=null)
{
        mp.prepare();
        mp.start();
        …
        mp.stop(streamID);
        mp.release();
        mp = null;
}
```