Mobil megoldások

Kocsis Gergely

2019.09.16.

Szenzor típusok

Mozgás érzékelők
Három tengely menti gyorsulás-, és elfordulásérzékelők.
TYPE_ACCELEROMETER, TYPE_GRAVITY, TYPE_GYROSCOPE, TYPE_ROTATION_VECTOR

Környezeti érzékelők
A környezeti paraméterek meghatározása
TYPE_TEMPERATURE/TYPE_AMBIENT_TEMPERATURE, TYPE_PRESSURE, TYPE_LIGHT, TYPE_RELATIVE_HUMIDITY

Pozíció érzékelők
Az eszköz fizikai elhelyezkedésének meghatározása
TYPE_MAGNETIC_FIELD, TYPE_ORIENTATION

Fontosabb osztályok

SensorManager
Egy szenzor service létrehozására szolgál. Ennek segítségével lehet elérni az eszköz szenzorait.

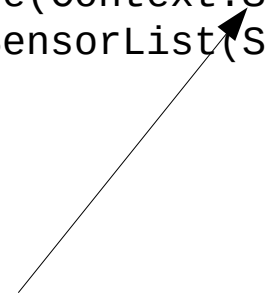
Sensor
Egy kifejezett szenzor példányosítására szolgál. Ezen keresztül éred el a szenzor által szolgáltatott értékeket.

SensorEvent
A szenzorral kapcsolatos változások detektálására szolgál. Az alábbiekat kérdezhetjük le tőle: nyers adat, gazda szenzor típusa, az adat pontossága, időbélyeg

SensorEventListener
Ezt az interfészt implementálva lehet elkapni a szenzor által kapott értékek változásait, vagy azok pontosságának változását

Szenzorok kiíratása

```
private SensorManager sensorManager;
...
sensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
List<Sensor> deviceSensors = sensorManager.getSensorList(Sensor.TYPE_ALL);

tv.setText(…);
```



Itt kérjük a Sensor Service elérését.
A Service-ek olyan Android alkalmazások / alkalmazás részletek, melyek nem rendelkeznek felhasználói felülettel és a háttérben képesek futni.
Nagyon sok ilyen érhető el. Nézzük meg.

A hivatalos példa osztály ServiceEvent-ek kezelésére

```
public class SensorActivity extends Activity implements SensorEventListener {
    private SensorManager sensorManager;
    private Sensor mLight;

    public final void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        sensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
        mLight = sensorManager.getDefaultSensor(Sensor.TYPE_LIGHT);
    }
    public final void onAccuracyChanged(Sensor sensor, int accuracy) {    }
    public final void onSensorChanged(SensorEvent event) {
        float lux = event.values[0];
    }
    protected void onResume() {
        super.onResume();
        sensorManager.registerListener(this, mLight, SensorManager.SENSOR_DELAY_NORMAL);
    }
    protected void onPause() {
        super.onPause();
        sensorManager.unregisterListener(this);
    }
}
```

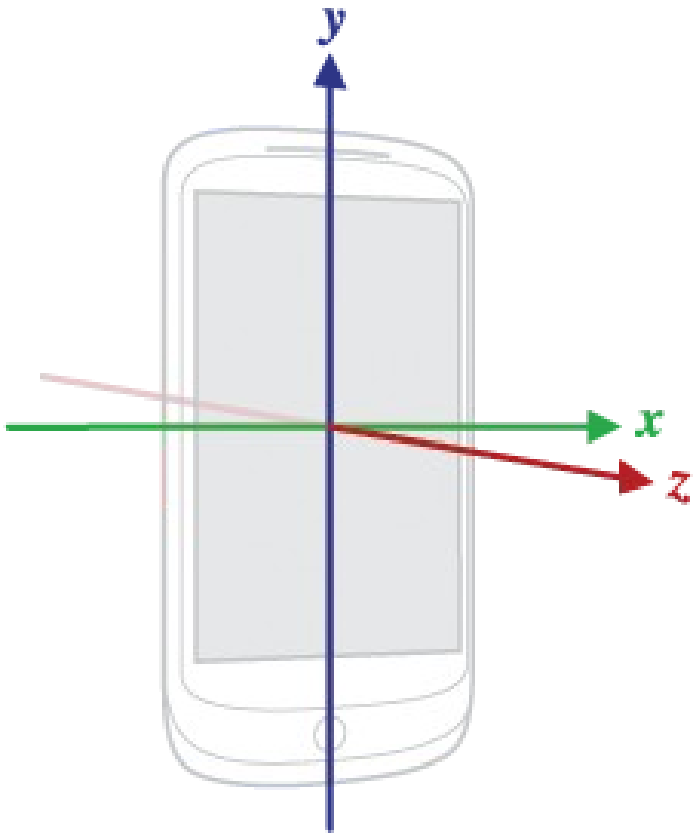
ServiceEvent-ek kezelése

```
public class MainActivity extends AppCompatActivity {

    private SensorManager sensorManager;
    private LightSensorEventListener sel = new LightSensorEventListener();
    private TextView tv;
    private Sensor mLight;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        sensorManager = (SensorManager)  getSystemService(Context.SENSOR_SERVICE);

        tv = findViewById(R.id.sensorList);
        tv.setMovementMethod(new ScrollingMovementMethod());
        mLight = sensorManager.getDefaultSensor(Sensor.TYPE_LIGHT);
        sensorManager.registerListener(sel, mLight, SensorManager.SENSOR_DELAY_NORMAL);
        sel.setTv(tv);
    }
    @Override
    protected void onResume() {
        super.onResume();
        sensorManager.registerListener(sel, mLight, SensorManager.SENSOR_DELAY_NORMAL);
    }
    @Override
    protected void onPause() {
        super.onPause();
        sensorManager.unregisterListener(sel);
    }
}
```

```
public class LightSensorEventListener implements SensorEventListener
{
    private TextView tv;
    public void setTv(TextView tv) {
        this.tv = tv;
    }
    @Override
    public void onSensorChanged(SensorEvent sensorEvent) {
        float lux = sensorEvent.values[0];
        tv.setText(""+lux);
    }
    @Override
    public void onAccuracyChanged(Sensor sensor, int i) {
        //TODO
    }
}
```

Háromtengelyű szenzorok

Az ilyen szenzorok esetén a tengelyek az ábrán látható módon értendők.
A tengelyek elosztása az eszközzel együtt mozog, nem fordul át a képrenyő elfordulásakor.



Rázás detektálása

Az Activity osztályba

```
// variables for shake detection
private static final float SHAKE_THRESHOLD = 3.25f; // m/S**2
private static final int MIN_TIME_BETWEEN_SHAKES_MILLISECS = 1000;
private long mLastShakeTime;
private SensorManager mSensorMgr;
```

Az onCreate metódusba

```
// Get a sensor manager to listen for shakes
mSensorMgr = (SensorManager) getSystemService(SENSOR_SERVICE);

// Listen for shakes
Sensor accelerometer = mSensorMgr.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
if (accelerometer != null) {
    mSensorMgr.registerListener(this, accelerometer,
SensorManager.SENSOR_DELAY_NORMAL);
}
```

Rázás detektálása

```
@Override
public void onSensorChanged(SensorEvent event) {
    if (event.sensor.getType() == Sensor.TYPE_ACCELEROMETER) {
        long curTime = System.currentTimeMillis();
        if ((curTime - mLastShakeTime) > MIN_TIME_BETWEEN_SHAKES_MILLISECS) {

            float x = event.values[0];
            float y = event.values[1];
            float z = event.values[2];

            double acceleration = Math.sqrt(Math.pow(x, 2) +
                    Math.pow(y, 2) +
                    Math.pow(z, 2)) - SensorManager.GRAVITY_EARTH;
            Log.d(APP_NAME, "Acceleration is " + acceleration + "m/s^2");

            if (acceleration > SHAKE_THRESHOLD) {
                mLastShakeTime = curTime;
                Log.d(APP_NAME, "Shake, Rattle, and Roll");
            }
        }
    }
}
```