Практична робота 10. Робота з датчиками та сервісами мобільного пристрою в Android Studio

Теоретичні відомості

Датчики, що стежать за фізичними властивостями та станом навколишнього середовища, надають інноваційні способи покращення мобільних додатків. Наявність у сучасних телефонах електронних компасів, датчиків рівноваги, яскравості та близькості відкриває цілу низку нових можливостей для взаємодії з пристроєм, таких як доповнена реальність та введення даних, що базується на переміщеннях у просторі.

Датчики в Android діляться на кілька категорій: руху, положення та навколишнього середовища. Нижче наведено деякі види популярних датчиків:

- Акселерометр (TYPE_ACCELEROMETER)
- Гіроскоп (TYPE_GYROSCOPE)
- Датчик освітлення (TYPE_LIGHT)
- Датчик відстані (TYPE_PROXIMITY)
- Датчик магнітних полів (TYPE_MAGNETIC_FIELD)
- Барометр (TYPE_PRESSURE)
- Датчик температури довкілля (TYPE_AMBIENT_TEMPERATURE)
- Вимірник відносної вологості (TYPE_RELATIVE_HUMIDITY)

Кожен телефон може мати свій набір датчиків. У більшості апаратів є акселерометр і гіроскоп.

Кожен з представлених датчиків заслуговує на окрему статтю. Майте на увазі, що існують застарілі класи для роботи з датчиками, зокрема для датчиків орієнтації та температури.

Необхідно пам'ятати кілька речей, працюючи з датчиками:

Показання бувають дуже нерівними. Вам потрібно використовувати якесь середнє значення свідчень, але не переборщити, щоб додаток залишався чуйним

Дані надходять нерівномірно. Не чекайте спокійного, рівного потоку даних

Спробуйте передбачити майбутні дії користувача. Наприклад, якщо дані про початок обертання пристрою, можна передбачити наступний рух і підготуватися до нього

На емуляторі практично неможливо тестувати роботу з датчиками, тому використовуйте реальні пристрої. В останніх версіях емуляторів перелік можливостей датчиків розширився. Дивіться в установках емулятора розділ Virtual sensors.

За роботу з сенсорами відповідає клас SensorManager, що містить кілька констант, що характеризують різні аспекти системи датчиків Android, у тому числі:

Тип датчика

Орієнтація, акселерометр, світло, магнітне поле, близькість, температура тощо.

Частота вимірів

Максимальна, для ігор, звичайна, для інтерфейсу користувача. Коли програма запитує конкретне значення частоти відліків, з погляду сенсорної підсистеми це лише рекомендація. Жодної гарантії, що вимірювання будуть проводитися із зазначеною частотою, немає.

Точність

Висока, низька, середня, ненадійні дані.

Типи датчиків

- TYPE_ACCELEROMETER - Вимірює прискорення у просторі по осях X, Y, Z
- TYPE_AMBIENT_TEMPERATURE - Новий датчик для вимірювання температури (API 14) у градусах Цельсія, який замінив застарілий TYPE_TEMPERATURE
- TYPE_GRAVITY - Триосьовий датчик сили тяжіння. Як правило, це віртуальний датчик і є низькочастотним фільтром для показань, що повертаються акселерометром

- TYPE_GYROSCOPE - Трьохосьовий гіроскоп, що повертає поточне положення пристрою в просторі в градусах по трьох осях. За іншими даними, повертає швидкість обертання пристрою за трьома осями в радіанах в секунду.
- TYPE_LIGHT – Вимірює ступінь освітленості. Датчик навколишнього освітлення, який описує зовнішню освітленість у люксах. Цей тип датчиків зазвичай використовується динамічного зміни яскравості екрана.
- TYPE_LINEAR_ACCELERATION - Триосьовий датчик лінійного прискорення, що повертає показники прискорення без урахування сили тяжіння. Це віртуальний датчик, який використовує показання акселерометра.
- TYPE_MAGNETIC_FIELD - Датчик магнітного поля, що визначає поточні показники магнітного поля в мікротесах по трьох осях.
- TYPE_ORIENTATION – Датчик орієнтації. Вимірює повороти, нахили та обертання пристрою. Вважається застарілим
- TYPE_PRESSURE – Датчик атмосферного тиску (барометр), що повертає поточний тиск у мілібарах. Можна визначати висоту над рівнем моря шляхом порівняння атмосферного тиску в двох точках. Також барометри можуть застосовуватись для прогнозування погоди.
- TYPE_PROXIMITY - Датчик наближеності, який сигналізує про відстань між пристроєм та цільовим об'єктом у сантиметрах. Яким чином вибирається об'єкт та які відстані підтримуються, залежить від апаратної реалізації даного датчика, можливе повернення двох значень – Близько та Далеко. Типове його застосування — Визначення відстані між пристроєм та вухом користувача для автоматичного регулювання яскравості екрана або виконання голосової команди.
- TYPE_RELATIVE_HUMIDITY - Датчик відносної вологості у вигляді відсоткового значення (API 14)

- TYPE_ROTATION_VECTOR - Повертає положення пристрою у просторі у вигляді кута щодо осі. Віртуальний датчик, що бере показання від акселерометра та гіроскопа. Також може використовувати показання датчика магнітного поля
- TYPE_GEOMAGNETIC_ROTATION_VECTOR – альтернатива TYPE_ROTATION_VECTOR. Найменша точність, але менша витрата батареї. З'явився в Android 4.4 (API 19)
- TYPE_POSE_6DOF – ще одна альтернатива TYPE_ROTATION_VECTOR. З'явився в Android 7.0 (API 24)
- TYPE_SIGNIFICANT_MOTION - З'явився в Android 4.3 (API 18)
- TYPE_MOTION_DETECT – детектор руху. З'явився в Android 7.0 (API 24)
- TYPE_STATIONARY_DETECT - З'явився в Android 7.0 (API 24)
- TYPE_STEP_COUNTER - датчик для підрахунку кількості кроків
- TYPE_STEP_DETECTOR - визначення початку кроків
- TYPE_HEART_BEAT – пульс. З'явився в Android 7.0 (API 24)
- TYPE_HEART_RATE – серцева активність. З'явився в Android 4.4 (API 20)
- TYPE_LOW_LATENCY_OFFBODY_DETECT - З'явився в Android 8.0 (API 26)

Крім апаратних датчиків, у пристроях використовуються віртуальні датчики, які надають спрощені, уточнені або комбіновані показання, використовуючи комбінацію з кількох апаратних датчиків. У деяких випадках цей спосіб є зручнішим.

Щоб отримати доступ до сенсорів, необхідно викликати метод getSystemService().

```
// Kotlin
private lateinit var sensorManager: SensorManager
sensorManager = getSystemService(SENSOR_SERVICE) as SensorManager
```

```
// Java
private SensorManager sensorManager;
sensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
```

Пристрій може включати кілька реалізацій одного і того ж типу датчиків. Щоб знайти стандартну реалізацію, викличте метод getDefaultSensor() з об'єкта SensorManager, передаючи йому як параметр тип датчика у вигляді однієї з констант, описаних вище.

Наступний фрагмент коду поверне об'єкт, який описує гіроскоп за замовчуванням. Якщо для цього типу немає датчика за промовчанням, буде повернуто значення null.

```
// Kotlin
sensorManager.getDefaultSensor(Sensor.TYPE_GYROSCOPE)
// Java
Sensor defaultGyroscope = sensorManager.getDefaultSensor(Sensor.TYPE_
GYROSCOPE);
```

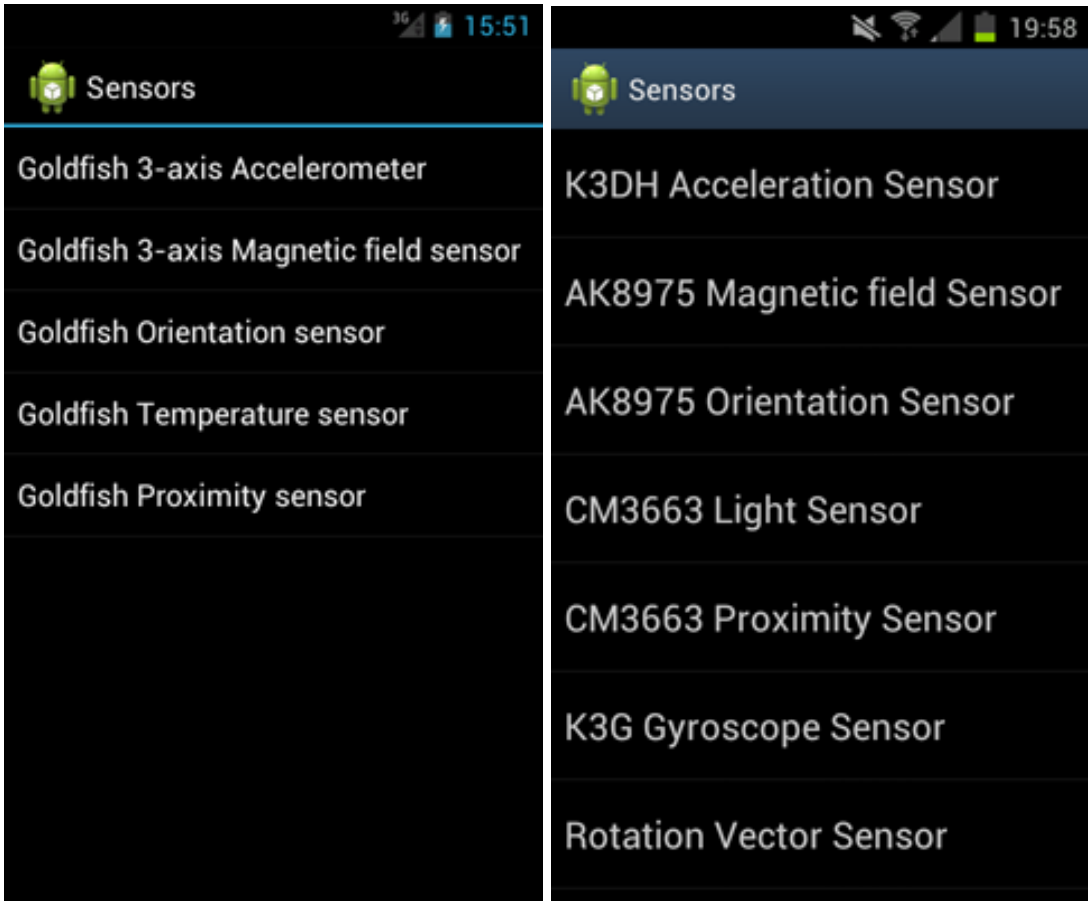
Клас SensorManager має метод getSensorList(), який дозволяє отримати список доступних датчиків на пристрої через константу Sensor.TYPE_ALL і метод getName():

```
// Kotlin
import android.hardware.Sensor
import android.hardware.SensorManager
import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity

class MainActivity : AppCompatActivity() {
    private lateinit var sensorManager: SensorManager
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
```

```
            sensorManager = getSystemService(SENSOR_SERVICE) as
SensorManager
            val deviceSensors: List<Sensor> = sensorManager.getSensorList(Sensor.
TYPE_ALL)
            println(deviceSensors.joinToString("\n"))
        }
    }
    // Java
    import java.util.ArrayList;
    import java.util.List;
    import android.app.ListActivity;
    import android.content.Context;
    import android.hardware.Sensor;
    import android.hardware.SensorManager;
    import android.os.Bundle;
    import android.widget.ArrayAdapter;

    public class SensorsActivity extends ListActivity {
        @Override
        protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            SensorManager sensorManager = (SensorManager) getSystemService(
Context.SENSOR_SERVICE);
            List<Sensor> deviceSensors = sensorManager.getSensorList(Sensor.
TYPE_ALL);
            List<String> listSensorType = new ArrayList<>();
            for (int i = 0; i < deviceSensors.size(); i++) {
                listSensorType.add(deviceSensors.get(i).getName());
            }
            setListAdapter(new ArrayAdapter<>(this,
```

```
                android.R.layout.simple_list_item_1, listSensorType));
        getListView().setTextFilterEnabled(true);
    }
}
```

Так як у кожного пристрою свій набір датчиків, результати будуть у всіх різними. Нижче наведені скріншоти з емулятора та реального пристрою. У першому випадку вивелося лише 5 датчиків, по-друге було набагато більше – ви бачите тільки ту частину, яка помістилася на екрані.



Також можна одержати список доступних датчиків конкретного типу. У наступному фрагменті коду будуть повернуті об'єкти Sensor, що являють собою всі доступні датчики тиску:

```
// Kotlin
val pressureSensors: List<Sensor> = sensorManager.getSensorList(Sensor.
TYPE_PRESSURE)
```

// Java

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

Також вам знадобиться інтерфейс android.hardware.SensorListener. Інтерфейс реалізований за допомогою класу, який використовується для введення значень датчиків у міру їхньої зміни в режимі реального часу. Додаток реалізує цей інтерфейс для моніторингу одного або кількох апаратних датчиків.

Інтерфейс включає два необхідні методи:

- Метод onSensorChanged(int sensor, float values[]) викликається щоразу, коли змінюється значення датчика. Цей метод викликається лише датчиків, контрольованих даним додатком. До аргументів методу входить ціле, яке вказує, що значення датчика змінилося, і масив значень з плаваючою комою, що відображають власне значення датчика. Деякі датчики видають лише одне значення даних, тоді як інші надають три значення з плаваючою комою. Датчики орієнтації та акселерометр дають по три значення даних кожен.
- Метод onAccuracyChanged(int sensor,int accuracy) викликається за зміни точності показань датчика. Аргументами служать два цілих числа: одне вказує датчик, інше відповідає новому значенню точності цього датчика.

Служба датчиків викликає onSensorChanged() щоразу при зміні значень. Усі датчики повертають масив значень із плаваючою точкою. Розмір масиву залежить від особливостей датчика. Датчик TYPE_TEMPERATURE повертає одне значення – температуру в градусах Цельсія, інші можуть повертати кілька значень. Ви можете використовувати лише потрібні значення. Наприклад, для отримання відомостей тільки про магнітний азімут достатньо використовувати перше число, яке повертається датчиком TYPE_ORIENTATION.

Параметр accuracy, що використовується в методах для представлення ступеня точності датчика, використовує одну з констант

- SensorManager.SENSOR_STATUS_ACCURACY_LOW. Говорить про те, що дані, що надаються датчиком, мають низьку точність і потребують калібрування.
- SensorManager.SENSOR_STATUS_ACCURACY_MEDIUM. Говорить про середній ступінь точності датчика і те, що калібрування може покращити результат.
- SensorManager.SENSOR_STATUS_ACCURACY_HIGH. Показники датчика точні настільки, наскільки це можливо.
- SensorManager.SENSOR_STATUS_UNRELIABLE. Дані, що надаються датчиком, є недостовірними. Це означає, що датчик необхідно калібрувати, інакше неможливо зчитувати результати.

Щоб отримувати події, що генеруються датчиками, зареєструйте свою реалізацію інтерфейсу SensorEventListener за допомогою SensorManager. Вкажіть об'єкт Sensor, за яким ви хочете спостерігати, та частоту, з якою вам потрібно отримувати оновлення.

Після отримання об'єкта ви викликаєте метод registerListener() у методі onResume(), щоб почати отримувати оновлені дані, і викликаєте unregisteredListener() метод onPause(), щоб зупинити отримання даних. У цьому випадку датчики будуть використовуватися лише тоді, коли активність помітна на екрані.

У наступному прикладі показаний процес реєстрації SensorEventListener для датчика наближеності за умовчанням із зазначенням стандартної частоти оновлення:

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

Клас SensorManager містить наступні константи для вибору відповідної частоти оновлень (у порядку зменшення):

- SensorManager.SENSOR_DELAY_FASTEST - найвища можлива частота оновлення показань датчиків;
- SensorManager.SENSOR_DELAY_GAME - частота, що використовується для управління іграми;
- SensorManager.SENSOR_DELAY_NORMAL - частота оновлень за замовчуванням;
- SensorManager.SENSOR_DELAY_UI — частота для оновлення інтерфейсу користувача.

Вибрана вами частота необов'язково дотримуватиметься. SensorManager може повертати результати швидше або повільніше, ніж ви зазначили (хоча, як правило, це відбувається швидше). Щоб мінімізувати витрати ресурсів при використанні датчиків у додатку, необхідно намагатися підбирати найнижчу частоту.

Android 7.0 Nougat (API 24) з'явилося поняття динамічних датчиків, розрахованих на платформу Android Things. Датчики можуть приєднуватися та від'єднуватись від плати у будь-який час.

Для визначення доступних динамічних датчиків використовуються методи isDynamicSensorDiscoverySupported(), isDynamicSensor(), getDynamicSensorList()

Момент приєднання чи від'єднання датчика від плати можна відстежувати через клас SensorManager.DynamicSensorCallback.

Завдання

Розробити мобільний додаток з використанням мови Java або Kotlin для використання датчиків мобільного пристрою (датчик визначається виколадачем).