Датчики

Датчики – это устройства для измерения сигналов от объекта и передачи этой информации на устройство. Современные телефоны содержат множество датчиков. Некоторые из них, такие как датчик приближения или гироскоп, являются обязательными, другие могут устанавливаться лишь в некоторые модели телефонов.

Список датчиков

Android поддерживает работу с множеством *аппаратных* датчиков, объявленных в классе `Sensor`:

- `TYPE_ACCELEROMETER`, `TYPE_LINEAR_ACCELERATION` – акселерометр, ускорение по трем осям, м/с^2;
- `TYPE_AMBIENT_TEMPERATURE` – температура окружающей среды, °C;
- `TYPE_GRAVITY` – сила тяжести по трём осям, м/с^2;
- `TYPE_GYROSCOPE` – гироскоп, скорость вращения устройства по трём осям, рад/с;
- `TYPE_HEART_BEAT` – детектор биения сердца;
- `TYPE_HEART_RATE` – пульс;
- `TYPE_LIGHT` – освещенность, лм;
- `TYPE_MAGNETIC_FIELD` – магнитометр, показания магнитного поля, мкТл;
- `TYPE_MOTION_DETECT`, `TYPE_STATIONARY_DETECT` – движение устройства: если устройство находится в движении (или в покое) от пяти до десяти секунд, возвращает 1;
- `TYPE_PRESSURE` – барометр, атмосферное давления, может использоваться для определения высоты, мБар;
- `TYPE_PROXIMITY` – приближенность, расстояние между устройством и пользователем, см, на некоторых устройствах возвращается только два значения: «далеко» и «близко»;
- `TYPE_RELATIVE_HUMIDITY` – относительная влажность, %;
- `TYPE_ROTATION_VECTOR` – вектор поворота;
- `TYPE_STEP_COUNTER` – количество шагов с момента включения устройства;
- `TYPE_STEP_DETECTOR` – срабатывает при каждом шаге пользователя.

Некоторые датчики, работающие с чувствительными данными о пользователях, требуют разрешений: например, датчик `TYPE_HEART_RATE` требует разрешения `BODY_SENSORS`, а датчики `TYPE_STEP_COUNTER` и `TYPE_STEP_DETECOR` требуют разрешения `ACTIVITY_RECOGNITION`.

При работе с аппаратными датчиками нужно помнить о следующих моментах:

- датчики работают без фильтрации или нормализации значений, поэтому показания бывают очень неровными: нужно усреднять значения, отбрасывая одиночные скачки показаний;
- данные приходят неравномерно: между показаниями могут быть перерывы разного интервала.

Некоторые датчики в Android могут быть *виртуальными*, они формируют свои значения программным способом, чтобы предоставить более сглаженные и точные результаты.

Примерами таких датчиков являются `TYPE_LINEAR_ACCELERATION` – высокочастотный, и `TYPE_GRAVITY` – низкочастотный фильтр для показаний акселерометра. На некоторых устройствах даже аппаратные датчики могут эмулироваться программным способом, например, датчик приближения `TYPE_PROXIMITY`.

Ряд датчиков со временем объявляются устаревшими, например, вместо датчика ориентации `TYPE_ORIENTATION`, который использовался для отслеживания поворотов, наклонов и вращения устройства, теперь рекомендуется использовать метод `SensorManager.getOrientation()`.

Получение показаний

Чтобы начать работу с датчиками, нужно получить системный менеджер датчиков:

```
val sensorManager = getSystemService(SENSOR_SERVICE) as SensorManager
```

Для получения списка датчиков используется метод `getSensorList`:

```
val sensors = sensorManager.getSensorList(Sensor.TYPE_ALL)
```

Полученный список будет включать все поддерживаемые датчики: как аппаратные, так и виртуальные. Для каждого датчика можно получить его свойства, например, название и режим срабатывания:

```
for (sensor in sensors) {
    Log.d("SENSORS", "name: ${sensor.name}")
    when (sensor.reportingMode) {
        Sensor.REPORTING_MODE_CONTINUOUS -> Log.d("SENSORS", "mode: CONTINUOUS")
        Sensor.REPORTING_MODE_ON_CHANGE -> Log.d("SENSORS", "mode: ON_CHANGE")
        Sensor.REPORTING_MODE_ONE_SHOT -> Log.d("SENSORS", "mode: ONE_SHOT")
        Sensor.REPORTING_MODE_SPECIAL_TRIGGER -> Log.d("SENSORS", "mode:
SPECIAL_TRIGGER")
    }
    Log.d("SENSORS", "")
}
```

Этот код выведет в панель LogCat примерно такие данные:

```
name: Goldfish Proximity sensor
mode: ON_CHANGE

name: Goldfish Light sensor
mode: ON_CHANGE

name: Goldfish Pressure sensor
mode: CONTINUOUS
```

Режим работы датчика может быть одним из следующих:

- `REPORTING_MODE_CONTINUOUS` – данные предоставляются постоянно;
- `REPORTING_MODE_ON_CHANGE` – данные предоставляются только при их изменении;
- `REPORTING_MODE_ONE_SHOT` – данные предоставляются один раз, после чего датчик деактивируется;

- `REPORTING_MODE_SPECIAL_TRIGGER` – предоставление данных может быть непостоянным, следует обратиться к документации по датчику для получения подробной информации.

Для получения списка всех доступных датчиков конкретного типа необходимо указать соответствующую константу:

```
val sensors = sensorManager.getSensorList(Sensor.TYPE_PRESSURE)
```

Аппаратные датчики помещаются в начало списка, виртуальные – в конец. Датчики могут различаться точностью, энергопотреблением и другими параметрами.

Вместо выбора датчика из списка проще попросить систему предоставить тот датчик, который выбран по умолчанию:

```
val sensor = sensorManager.getDefaultSensor(Sensor.TYPE_GRAVITY)
```

Функция сначала ищет подходящий аппаратный датчик, если такого нет – виртуальный, а если и он отсутствует – возвращает `null`.

Информация от датчика приходит в слушатель, реализующий интерфейс `SensorEventListener`:

```
var listener = object : SensorEventListener {

    override fun onAccuracyChanged(sensor: Sensor, accuracy: Int) {}

    override fun onSensorChanged(event: SensorEvent) {
       if (event.sensor.type == Sensor.TYPE_GRAVITY)
          Log.d("DATA", "${event.values[0]} : ${event.values[1]} : ${event.values[2]}")
    }
}
```

Интерфейс требует реализации двух функций:

- `onAccuracyChanged` – информирует об изменении точности передаваемых данных:
  - `SENSOR_STATUS_ACCURACY_HIGH` – высокая точность;
  - `SENSOR_STATUS_ACCURACY_MEDIUM` – средняя точность, калибровка может улучшить точность;
  - `SENSOR_STATUS_ACCURACY_LOW` – низкая точность, требуется калибровка;
  - `SENSOR_STATUS_UNRELIABLE` – данные недостоверны, требуется калибровка;
  - `SENSOR_STATUS_NO_CONTACT` – данные не получены потому что у датчика нет контакта с тем, что он измеряет.
- `onSensorChanged` – информирует о поступлении данных от датчика.

Данные, сформированные датчиком, передаются в объекте `event.values`, который представляет собой массив значений с плавающей точкой `FloatArray`. Количество значений в массиве может быть разным в зависимости от того, какой именно датчик их прислал: например, гироскоп возвращает 3 числа, показывающие скорость вращения по каждой из трёх осей в радианах в секунду, а датчик освещённости – только 1 число, показывающее степень освещённости в люксах.

Когда требуется начать получать данные от датчика, нужно зарегистрировать слушатель таким образом:

```
sensorManager.registerListener(listener, sensor, SensorManager.SENSOR_DELAY_NORMAL)
```

Первый параметр представляет собой слушателя, второй – объект датчика, а третий – как часто приложение желает получать данные от датчика:

- `SENSOR_DELAY_FASTEST` – максимальная частота обновления данных (задержка 0 мс);
- `SENSOR_DELAY_GAME` – частота для игр с использованием гироскопа (задержка 20 мс);
- `SENSOR_DELAY_NORMAL` – частота обновления по умолчанию;
- `SENSOR_DELAY_UI` – частота, подходящая для обновления пользовательского интерфейса (задержка 60 мс).

Впрочем, система не всегда соблюдает заявленные требования к частоте обновления.

Если необходимости в показаниях датчика больше нет, следует отменить регистрацию:

```
sensorManager.unregisterListener(listener)
```

Обычно регистрация отменяется также при уходе активности с экрана, например, в обработчике события `onPause`.

Задание

Разработайте приложение, которое рисует на экране шарик, подчиняющийся закону гравитации. Для измерения гравитации используйте датчик `TYPE_GRAVITY`. При изменении положения телефона в пространстве будут изменяться и показатели гравитации, действующие вдоль каждой из осей. Используйте эти данные для расчёта скорости перемещения шарика на экране.

Датчик `TYPE_GRAVITY` возвращает три значения. Если телефон лежит на столе экраном вверх, то вот какая реакция будет на изменение его положения:

- Приподнимаем со стола **левый** край – значение **0** уменьшается от 0 к -9.8
- Приподнимаем со стола **правый** край – значение **0** увеличивается от 0 к +9.8
- Приподнимаем со стола **верхний** край – значение **1** увеличивается от 0 к +9.8
- Приподнимаем со стола **нижний** край – значение **1** уменьшается от 0 к -9.8

Попробуйте подключить датчик и организовать вывод значений в LogCat, а затем начать изменять положение телефона в пространстве, посмотрите, как будут меняться значения по каждой из трёх осей.

Немного физики и формул

Для расчёта позиции шарика на экране могут понадобиться некоторые формулы. Центр шарика размещается на экране в координатах [x,y], у него есть скорости перемещения по каждой из осей, назовём их Vx и Xy. Из физики известна формула ускорения при равноускоренном движении:

$$a = \frac{V - V_0}{t}$$

Здесь a – ускорение, V – скорость в текущий момент времени, V_0 – предыдущая скорость, t – прошедшее время.

Во всех последующих расчётах будет логично принять t равным 1, поскольку при очередном получении данных от датчика будет рассчитываться изменение значений по сравнению с предыдущим шагом.

Тогда при получении новых значений ускорения, новая скорость движения шарика по выбранной оси будет рассчитываться так:

$$V = V_0 + a$$

Чтобы определить новые координаты шарика, можно просто прибавить к текущей координате скорость:

$$S = S_0 + V$$

Здесь S_0 – текущая координата шарика (x или y), V – текущая скорость.

Таким образом, когда с датчика приходят очередные данные, нужно изменить в соответствии с ними скорости по каждой из осей, а затем изменить координаты, прибавив к ним скорость.

Когда шарик долетает до одной из границ экрана, нужно чтобы он отскочил. Для этого скорость меняется на противоположную, с помощью смены знака с плюса на минус или обратно. Нужно помнить, что у шарика есть радиус, его тоже необходимо учитывать при определении долёта шарика до границы экрана.