ЛЕКЦІЯ 11. ДАТЧИКИ ТА СЕНСОРНИЙ ЕКРАН

Планшетні комп'ютери, як і багато інших мобільні пристрої мають набір датчиків, наприклад, датчиком орієнтації, освітлення, акселерометром і т.п. Для підтримки такого різнорідного набору датчиків потрібно самостійний API. Датчики, в свою чергу, дозволяють реалізувати більш зручний, більш інтуїтивний інтерфейс управління програмою на планшеті.

Датчики - невід'ємна частина сучасного планшета або смартфона. У телефонах вони з'явилися дуже давно, наприклад, датчик, що дозволяє телефону визначити, прикладений він до вуха або повинен працювати в режимі гучного зв'язку.

Зараз кількість датчиків і їх можливості суттєво зросли. Це стосується як датчиків високого рівня отримання поточної орієнтації екрану (портрет, пейзаж), так і низького рівня, як, наприклад, отримання в режимі реального часу показань акселерометра.

Разом з тим набір датчиків змінюється від пристрою к пристрою, більш того, важко передбачити, які нові датчики з'являться в майбутньому. Дані, отримані від датчиків, неоднорідні. Ці обмеження роблять API датчиків щодо складним.

Сенсорні можливості Android

Один з приємних аспектів роботи з платформою Android полягає в можливості отримати доступ до деяких корисним компонентам самого пристрою. Досі розробників мобільних пристроїв засмучувала неможливість доступу до їх внутрішнього обладнання. Хоча між програмістом і металом все ж залишається Java-середовище Android.

Команда розробників Android вивела багато можливостей апаратури на поверхню. А так як Android - платформа з відкритим вихідним кодом, то можна засукати рукава і написати власний код для вирішення своїх завдань. В Android існує багато різних датчиків, використовуючи які, розробники можуть створювати цікаві і корисні програмні рішення.

Інтерфейс традиційних датчиків в Android API

Розглянемо більш докладно роботу з датчиками Android. Пакет android.hardware представляє розробнику API, який може бути використаний при необхідності в додатках, спираються на апаратні можливості пристрою.

Наприклад, пакет являє інтерфейс управління камерою і іншими датчиками пристрою. Операційна система Android за умовчанням програмну абстракцію будь-якого фізичного елемента пристрою, будь то камера або датчик руху, але програмуючи додатки для роботи з датчиками пристрою, необхідно спочатку переконається, що необхідний датчик присутній в мобільному пристрої. щоб убезпечити себе від таких помилок потрібно в маніфест файлі програми використовувати директиву <uses-feature>.

Розглянемо опис апаратно-орієнтованих інтерфейсів Android.

Camera.AutoFocusCallback - інтерфейс, дозволяє довідуватися про закінчення автофокусування камери. Реєструє подія на програмному рівні ОС.

Camera.ErrorCallback - інтерфейс, дозволяє довідуватися про помилки. Реєструє події на програмному рівні ОС.

Camera.FaceDetectionListener - інтерфейс, дозволяє розпізнавати обличчя в попередньому перегляді. Реєструє події на програмному рівні ОС.

Camera.OnZoomChangeListener - інтерфейс, дозволяє довідуватися про зміни зуму камери. Реєструє події на програмному рівні ОС.

Camera.PictureCallback - інтерфейс зворотного, виклику використовується для подачі даних після зйомки. Інтерфейс зворотного виклику Camera.PreviewCallback використовуються, щоб надати копії попереднього перегляду кадрів як вони відображаються на пристрої.

Camera.ShutterCallback - інтерфейс зворотного виклику використовується для позначення моменту фактичного захоплення зображення.

SensorEventListener - інтерфейси використовується для отримання повідомлень від менеджера датчиків (SensorManager), в той момент, коли значення датчика змінилося.

SensorListener - інтерфейс реалізований за допомогою класу, котрий використовується для виведення значень датчиків по міру їх зміни в режимі реального часу. Додаток реалізує цей інтерфейс для моніторингу одного або декількох наявних апаратних датчиків.

Camera - клас камери, використовується для установки настройки захоплення зображення, старт/стоп, попереднього перегляду, фотографії і вилучення кадрів для кодування відео.

Sensor – касс представляє датчик. Клас представляє датчик події, а SensorEvent також містить корисну інформацію таку, як тип сенсора, тимчасова мітка, точність і дані сенсора.

SensorManager - класс, забезпечує доступ до внутрішніх датчиків платформи Android.

Пакет android.os. * пакет, що містить кілька корисних класів для взаємодії з операційної середовищем, включаючи управління харчуванням, пошук файлів, обробник і класи для обміну повідомленнями. Як і багато інших пристроїв телефони на базі Android можуть споживати досить багато електроенергії. Забезпечення роботи пристрою в потрібний момент, щоб проконтролювати потрібну подію, це важливий аспект проектування.

SensorEventLstener. Використовується для отримання повідомлень від SensorManager, в той момент, коли показання датчика змінюються. Містить опис двох основних методів, які необхідно описати в класі.

```
1) public abstract void onAccuracyChanged (Sensor sensor, int accuracy);
```

Викликається тоді, коли точність показань датчика змінюється. Параметр sensor визначає датчик - об'єкт класу Sensor. Параметр accuracy визначає точність вимірювань датчика. Точність може бути: висока, низька, середня, ненадійні дані.

```
2) public abstract void onSensorChanged (SensorEvent event, float values[]);
```

Метод викликається щоразу, коли змінюється значення датчика. Цей метод викликається тільки для датчиків контрольованих самим додатком. У число аргументів методу входить ціле число, яке вказує, що значення датчика змінилося, та масив значень з плаваючою комою, які відображають значення датчика.

Деякі датчики видають тільки одне значення даних, тоді як інші надають три значення з плаваючою коми. Датчики орієнтації і акселерометр дають по три значення даних кожен.

SensorManager. Надає доступ до різним датчикам пристрою. використовуючи метод getSystemService з параметром SENSOR_SERVICE, можна, можливо отримати екземпляр класу.

Програмуючи додатки, використовують датчики завжди необхідно переконатися в тому, що датчики не функціонує, коли додаток призупинено.

Приклад демонструє основи роботи з класом SensorManager і Sensor. Необхідний датчик може бути отриманий за допомогою виклику методу, об'єкта класу SensorManager, getDefaultSensor (Sensor. <тип>).

Всі типи датчиків Android описані у вигляді констант в класі Sensor. Методи onResume () і onPause () необхідні для економічного використання енергоресурсів пристрою.

Всякий раз, коли додаток призупиняє свою роботу, скидайте слухач датчика. для взаємодії з датчиком додаток має зареєструватися на прийом дій, пов'язаних з одним або декількома датчиками. Реєстрація здійснюється з допомогою методу SensorManager.registerListener ().

SensorEvent. Клас описує різні типи подій датчиків, які обчислюються пристроєм по різному в залежно від різновиду датчика. Будучи шаблоном, описує процес функціонування різних датчиків (прискорення, орієнтації...), клас представляє собою дуже гнучкий засіб отримання показань від цих компонентів пристрою. Розглянемо більш докладно структуру класу.

Клас має чотири основних поля, які розкривають процес взаємодії датчика пристрою з навколишнім світом.

1) Accuracy - поле визначає точність показань сенсора, зазвичай значення цієї величини - константа, яка задається при реєстрації слухача на об'єкт класу Sensor.

2) Sensor - об'єкт згенерував подія.

3) Timestamp - величина типу long, що повідомляє час (в наносекундах) виникнення події.

4) Float values[] - значення з датчика, які відображають процес взаємодії пристрою з навколишнім світом. цей параметр залежить від типу датчика.

```
public class SensorActivity extends Activity, implements SensorEventListener {
```

```
private final SensorManager mSensorManager; private final Sensor mAccelerometer;
public SensorActivity () {mSensorManager =
(SensorManager) getSystemService (SENSOR_SERVICE); mAccelerometer =
mSensorManager.getDefaultSensor
(Sensor.TYPE_ACCELEROMETER);
}
protected void onResume () {super.onResume ();
mSensorManager.registerListener (this, mAccelerometer,
SensorManager.SENSOR_DELAY_NORMAL);
}
protected void onPause () {super.onPause ();
mSensorManager.unregisterListener (this);
}
public void onAccuracyChanged (Sensor sensor, int accuracy) {
}
public void onSensorChanged (SensorEvent event) {
}
}
```

Датчик орієнтації

Пристрій з попередньо встановленою ОС Android включає датчик орієнтації, який використовується для розпізнавання положення телефону в просторі. Як трактуються координатні осі в Android продемонстровано на рис. 1.

Орієнтація в Android визначається трьома величинами:

1. Азимут в градусах - кут між віссю X і північним напрямком 0 <= azimuth <= 360.

2. Висота в градусах - кут між віссю У і горизонтальним становищем. -180 <= Pitch <= 180.

3. Обертання в градусах - кут між віссю X і горизонтальним становищем приладу. -90 <= Roll <= 90.

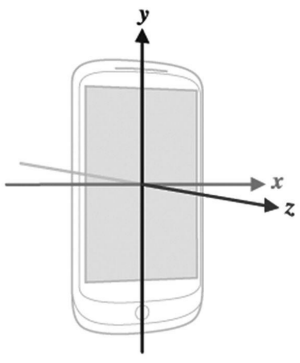

Рис. 1. Напрямок координатних осей в датчику орієнтації

Необхідно використовувати метод getSystemService (Context. SENSOR _ SERVICE) щоб форматувати SensorManager. Далі getDefaultSensor(Sensor.TYPE_ ORIENTATION) щоб створити об'єкт класу Sensor і форматувати його як датчик орієнтації.

Для роботи з датчиком орієнтації необхідно реалізовувати два методу, оголошені в інтерфейсі: перший перевіряє зміну точності, другий викликається всякий, раз коли відбувається зміна показань датчика.

Датчик акселерації

Android підтримує велику кількість датчиків, які можуть бути використані для отримання інформації. Зараз ми познайомимося, як в додатку використовувати датчик прискорення.

Для того, щоб використовувати датчик прискорення, в програмах необхідно реалізовувати той ж інтерфейс, що був описаний вище. Параметр rate методу registerlistener() дозволяє задавати точність показань датчика прискорення пристрою.

Необхідно використовувати метод getSystemService(Context.SENSOR_ SERVICE) щоб форматувати SensorManager. Далі використовується метод getDefaultSensor (Sensor. TYPE _ ACCELEROMETER) щоб створити об'єкт класу Sensor і форматувати його як датчик акселерації.

Розглянемо поля об'єкта події методу onSensorChanged (SensorEvent):

int accuracy - визначає точність вимірювань.

long timestamp - час в наносекундах, ідентифікує початок події.

float [] values - значення в системі CI.

values [0] - поточне прискорення по X, мінус прискорення вільного падіння по осі X.

values [1] - поточне прискорення по Y, мінус прискорення вільного падіння по осі Y.

values [2] - поточне прискорення по Z, мінус прискорення вільного падіння по осі Z.

Датчик GPS

Android дозволяє використовувати показання GPS-датчика в тих випадках, коли логіка додатки безпосередньо залежить від розташування пристрою в просторі, щодо Землі.

GPS-датчик не є стандартним датчиком в Android, тому для його використання застосовується трохи інший підхід.

LocationManager - менеджер управління службою GPS, основний клас, який представляє GPS-датчик в Android. Клас надає доступ до системної службі локації, що дозволяє додатком отримувати періодичні поновлення в протягомдеякого проміжку часу.

Для ініціалізації об'єкта класу LocationManager використовується метод getSystemService () з параметром Location_Service.

Щоб запустити службу GPS використовуйте метод requestLocationUpdate (String provider, long minTime, float minDistance, LocationListener listener).

Параметр provider задається в вигляді статичної константи, яка визначає постачальника послуги GPS. Наприклад, якщо вказати LocationManager.GPS_PROVIDER, в якості параметра, то постачальником послуги буде супутник GPS, якщо вказати LocationManager.NETWORK_PROVIDER - то, постачальником послуги буде мережевий протокол UDP або HTTP.

Другий параметр задає періодичність отримання даних від GPS служби. Третій параметр визначає мінімальний розмір області простору, в якому ми не хочемо отримувати повідомлення від GPS служби. Цей параметр визначається логікою програми.

Четвертий параметр - слухач, який реагує на зміни локації.

Для того, щоб скинути слухач використовуйте метод removeUpdates (LocationListener listener).

Розглянемо докладно інтерфейс LocationListener. Він включає в себе такі методи:

1) onLocationChanged (Location location) - метод викликається всякий, раз коли відбувається зміна показань GPS датчика. Кількість викликів даного методу безпосередньо залежить від того, як ви на нього підписалися (параметри

minDistance, minTime). Екземпляр класу Location містить поточне значення датчика. Його основні поля: londitude (довгота), latitude (широта), altitude (висота над рівнем моря), accuracy (точність), timestamp (час ідентифікації події).

2) onStatusChanged (String provider, int status, Bundle extras) - метод викликається щоразу, коли GPS програмно терпить зміни (поганий сигнал, пристрій не відповідає і. т. д). Наприклад даний метод викликається, коли GPS служба не в змозі встановити місце розташування або недавно стала доступною після періоду блокування.

3) onProviderEnabled (String provider) - метод визивається, коли програмі доступна служба GPS, або визивається після того, як користувач включить GPS.

4) onProviderDisabled (String provider) - метод викликається, коли програмі недоступний GPS, взагалі кажучи, якщо пристрій ніколи не використовувало службу GPS, то цей метод після автоматичного виклику повинен надати користувачеві можливість включити GPS.