akademia androida

Sensory

część V

agenda

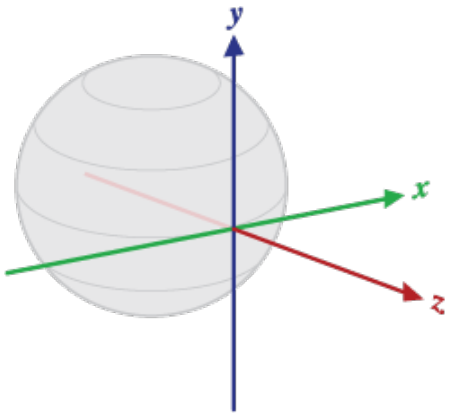
1. O sensorach słów kilka
2. Sensor Framework
3. Akcelerometr
4. Czujnik zbliżeniowy
5. Czujnik światła
6. Zadanie 1.
7. Zadanie 2 (domowe)

1. O sensorach słów kilka

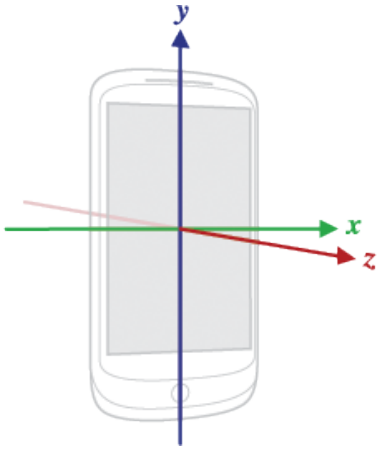
Każdy smartfon wyposażony w Androida ma również na pokładzie jakiś sensor. Sensory ułatwiają nam komunikację z urządzeniem, czyniąc użytkowanie przyjemniejszym. System Android wspiera różne rodzaje sensorów, które możemy podzielić na 3 grupy:

- sensory ruchu
- sensory pozycji
- sensory środowiskowe

Sensory ruchu są odpowiedzialne za pomiary sił przyspieszenia i sił obrotowych zachodzących wzdłuż 3 osi: X, Y i Z. Są to osie bezwzględne, niezależne od urządzenia. Do tej kategorii sensorów możemy zaliczyć np. akcelerometr czy żyroskop.



Sensory pozycji są odpowiedzialne za pomiary związane z fizycznym położeniem urządzenia. Osie są zdefiniowane względnie do ekranu w jego domyślnej pozycji. Do tej kategorii należą sensory położenia i magnetometr.



Sensory środowiskowe są odpowiedzialne za pomiary różnych parametrów związanych z otaczającym urządzenie środowiskiem, np. stopnia oświetlenia, wilgotność, ciśnienie czy temperaturę powietrza. Do tej kategorii zaliczymy barometr, fotometr (czujnik oświetlenia) i termometr (o ile dane urządzenie je posiada).

W zależności od budowy sensora możemy je podzielić na:

- sprzętowe
- programowe

2. Sensor Framework

Klasa Sensor

Pozwala na utworzenie instancji konkretnego sensora.

By dowiedzieć się, jakimi sensorami dysponuje nasz smartfon możemy użyć metody `getSensorList(int)`, która zwróci nam listę wykrytych sensorów pewnego typu. Gdy podamy jako parametr `Sensor.TYPE_ALL` lista będzie zawierać wszystkie sensory różnych typów.

Przykład użycia:

```
List<Sensor> deviceSensors = mSensorManager.getSensorList(Sensor.TYPE_ALL);
```

Lista dostępnych sensorów wspieranych przez API Androida:

TYPE_ACCELEROMETER
TYPE_ALL
TYPE_AMBIENT_TEMPERATURE
TYPE_GRAVITY
TYPE_GYROSCOPE
TYPE_LIGHT
TYPE_LINEAR_ACCELERATION
TYPE_MAGNETIC_FIELD
TYPE_ORIENTATION
TYPE_PRESSURE
TYPE_PROXIMITY
TYPE_RELATIVE_HUMIDITY
TYPE_ROTATION_VECTOR
TYPE_TEMPERATURE

Więcej info:
http://developer.android.com/reference/android/hardware/Sensor.html

Klasa SensorEvent

Reprezentuje wszystkie zdarzenia związane z danym sensorem i zawiera informacje takie jak typ sensora, jego dokładność i oczywiście dane odczytane przez sensor. Zawiera także definicję układu współrzędnych urządzenia.

Klasa SensorManager

Aby uzyskać dostęp do sensorów urządzenia potrzebna jest referencja do `sensor service`. Referencję uzyskamy poprzez utworzenie obiektu klasy `SensorManager` i wywołanie metody `getSystemService()` :

```
SensorManager mSensorManager = (SensorManager)getSystemService(SENSOR_SERVICE);
```

Samo utworzenie obiektu to jednak nie wszystko. By aplikacja wykorzystywała konkretny sensor, musimy jej „powiedzieć" o który nam chodzi. Do tego istnieje metoda `getDefaultSensor()`, a żeby zacząć korzystać z niego posłużyć się metodą:

```
registerListener (SensorEventListener listener, Sensor sensor, int rate);
```

którą wywołujemy na obiekcie klasy `SensorManager`.

Ważną rzeczą przy używaniu sensorów jest **wyłączenie** dostępu do nich, a z których korzystaliśmy w trakcie działania aktywności. Zapominając o tym nasza bateria padnie dość szybko. Do wyłączenia konkretnego sensora służy metoda `unregisterListener()`, która jest wywoływana na obiekcie klasy `SensorManager`.

Interfejs SensorEventListener

Interfejs implementujący dwie metody: `onAccuracyChanged()` i `onSensorChanged()` które pozwalają odbierać dane z sensorów, gdy zmieni się jego dokładność lub położenie.

Dobre nawyki, które należy sobie wyrobić przy używaniu sensorów:

1. Gdy aplikacja ma kończyć lub zawieszać swe działanie należy zwalniać uchwyt sensora (`unregisterListener()`)
2. Nie testować aplikacji na emulatorze
3. Nie blokować metody `onSensorChanged()` !!!
4. Starać się unikać używania przestarzałych metod czy sensorów
5. Sprawdzać istnienie sensorów przed ich użyciem

3. Akcelerometr

Przyrząd do pomiaru przyspieszeń liniowych lub kątowych. Mierzy swój własny ruch. Obecnie coraz częściej instalowane w mobilnych urządzeniach elektronicznych umożliwiając automatyczne wykrywanie ułożenia przestrzennego urządzenia oraz sterowanie jego funkcjami przez poruszanie nim.

Dostęp do niego uzyskamy w sposób:

```
private SensorManager mSensorManager;
private Sensor mSensor;
  ...
mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
mSensor = mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
```

Jak poprawnie uzyskać dostęp do akcelerometru:

```
protected void onResume() {
    super.onResume();
    mSensorManager.registerListener(this, mSensor,
SensorManager.SENSOR_DELAY_NORMAL);
}
protected void onPause() {
   super.onPause();
   mSensorManager.unregisterListener(this);
}
public void onSensorChanged(SensorEvent event) {
//co chcemy zrobić z danymi odczytanymi przez akcelerometr
}
```

Tip: Wartości odczytane przez akcelerometr (oś Z X Y) są umieszczone w tablicy. Można się do nich dostać tak: `event.values[indeks]`.

4. Czujnik zbliżeniowy

Rodzaj czujnika, który reaguje na zbliżanie się obiektu. Reakcja czujnika rozpoczyna się, gdy zbliżający się obiekt przekroczy pewien próg. Urządzenie najczęściej zwraca wartości 0 lub 1 – odpowiadają stanom blisko – daleko.

Dostęp do czujnika zbliżeniowego uzyskamy w sposób:

```
private SensorManager mSensorManager;
private Sensor mSensor;
...
mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
mSensor = mSensorManager.getDefaultSensor(Sensor.TYPE_PROXIMITY);
  public void onSensorChanged(SensorEvent event) {
    float distance = event.values[0];
    // co chcemy zrobić z danymi odczytanymi przez akcelerometr
  }
  protected void onResume() {
    super.onResume();
    mSensorManager.registerListener(this, mSensor,
SensorManager.SENSOR_DELAY_NORMAL);
  }
  protected void onPause() {
    super.onPause();
    mSensorManager.unregisterListener(this);
  }
```

5. Czujnik światła

Czujnik reagujący na zmianę intensywności docierającego do niego strumienia światła. Może być wykorzystywane zarówno światło widzialne jak również podczerwone lub laserowe.

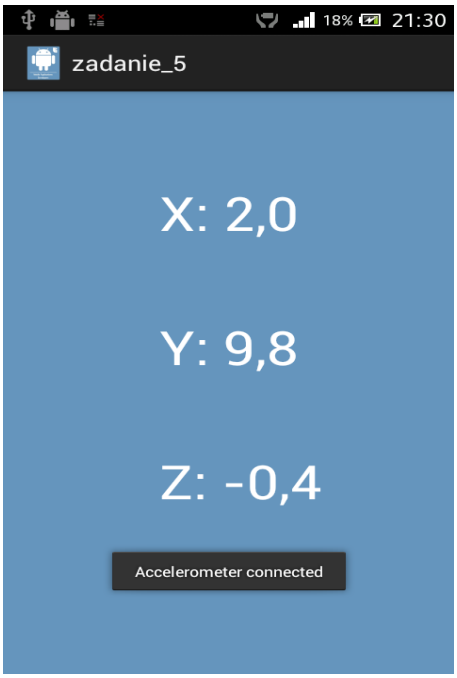
Dostęp do czujnika światła:

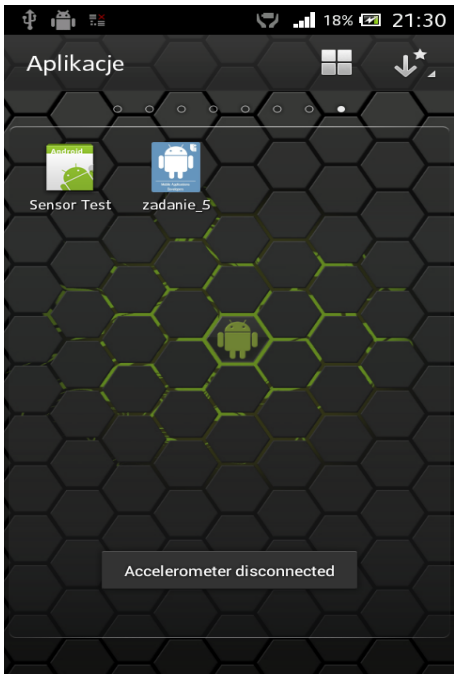
```
private SensorManager mSensorManager;
private Sensor mSensor;
...
mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
mSensor = mSensorManager.getDefaultSensor(Sensor.TYPE_LIGHT);
```

Tip: po każdym utworzeniu uchwytu warto sprawdzić, czy jego wartość nie jest równa `null`. Będziemy mogli wtedy zdefiniować jakąś akcję/powiadomienie w przypadku braku takiego sensora.

6. Zadanie 1.

- Utworzyć 3 pola `TextView`
- Uzyskać uchwyt do akcelerometru
- W metodzie `onSensorChanged()` wyświetlić dane odczytane z sensora dla każdej osi
- Wyświetlane dane muszą być sformatowane do jednego miejsca po przecinku
- W `AndroidManifest.xml` wymusić orientację ekranu na `portrait`
- W odpowiednich metodach (`onCreate()`, `onPause()`, `onResume()`)dodać powiadomienia `Toast`, sprawdzając tym samym, czy uchwyt do sensora działa
- **Pamiętać o zwolnieniu sensora w odpowiedniej metodzie**





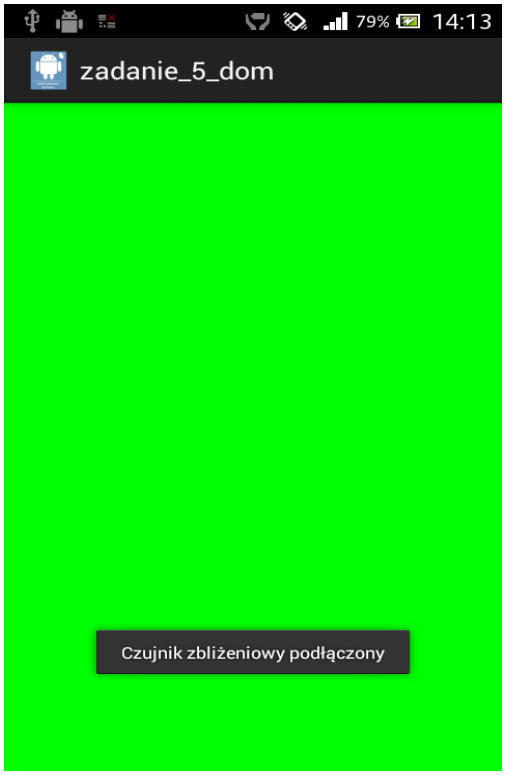
7. Zadanie 2 (domowe)

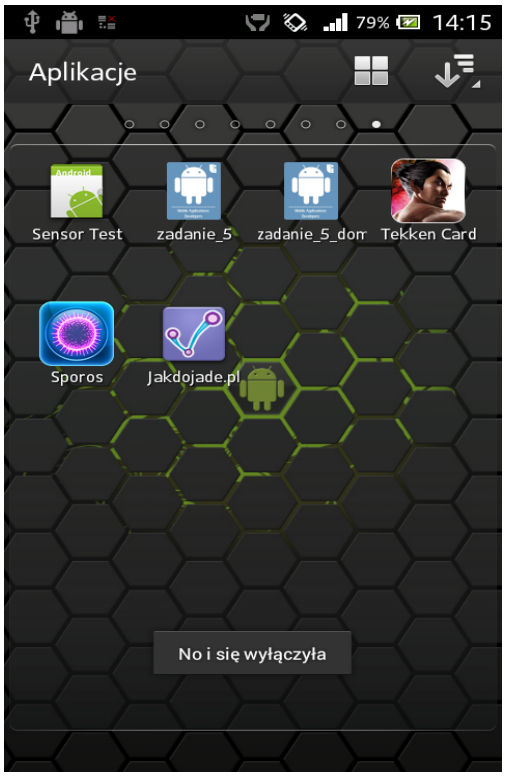
- W aktywności utworzyć obiekt klasy `View`
- Należy ustawić kolor tła na **zielony**
- Uzyskać uchwyt do czujnika zbliżeniowego
- W `onSensorChanged()` wywołać własną metodę przyjmującą za parametr zmienną float
- We własnej metodzie zdefiniować akcję dla zbliżenia:
  - każde zbliżenie ma być zliczane
    - w Toast ma zostać wyświetlona informacja o pozostałej ilości zbliżeń
    - po 5 zbliżeniach aktywność kończy działanie (Toast o tym zdarzeniu)
    - przy przedostatnim zbliżeniu użytkownik ma dostać informację o

zbliżającym się zamknięciu aplikacji

- każde zbliżenie zmienia kolor ekranu na **czerwony**

- W odpowiednich metodach (`onCreate()`, `onPause()`, `onResume()`)dodać powiadomienia `Toast`, sprawdzając tym samym, czy uchwyt do sensora działa
- **Pamiętać o zwolnieniu sensora w odpowiedniej metodzie**





Dziękuję za uwagę

kontakt:

- dglinski@wi.zut.edu.pl
- mad@zut.edu.pl