Proiect - Achiziții de date și instrumentație virtuală

Autor: Bloțiu Mihnea-Andrei - 323CA

Universitatea Politehnica București, Facultatea de Automatică și Calculatoare
mihnea.blotiu@stud.acs.upb.ro

Abstract. În acest proiect, voi explica pașii urmați crearea unui joc funcțional pe sistemul de operare - android. Jocul prezintă o bilă ce se mișcă pe ecran în funcție de orientarea și respectiv, înclinația telefonului.

Keywords: Android · Accelerometru · Orientare · Android Studio - Java.

1 Introducere

1.1 Scopul proiectului

Scopul principal al acestei aplicații a fost accesarea și utilizarea a unui senzor din multitudinea prezentă pe un dispozitiv Android și anume accelerometrul prin crearea unui joc simplu ce îl utilizează.

1.2 Accelerometrul - Explicații generale

Accelerometrele din telefoanele mobile sunt folosite pentru a detecta orientarea telefonului.

Un accelerometru măsoară accelerația liniară a mișcării. În practică, asta înseamnă că un accelerometru va măsura mișcarea direcțională a unui dispozitiv, dar nu va putea rezolva cu precizie orientarea laterală sau înclinarea acestuia în timpul mișcării, decât prin prezența unui giroscop ce va completa aceste informații.

Cu un accelerometru putem obține de obicei, o ieșire de informații cu adevărat „zgomotoasă". Pentru a evita „zgomotul" și a obține o ieșire clară, dar și rapidă, adică în timp util, accelerometrul cu 3 axe trebuie combinat cu un giroscop cu 3 axe.

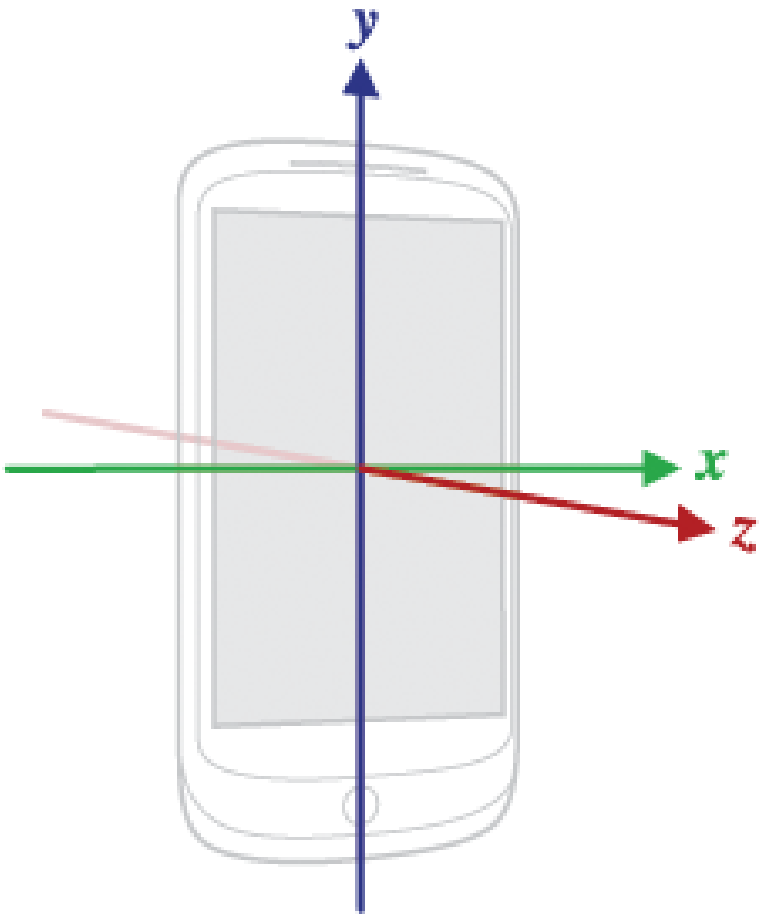

Fig. 1. Reprezentarea grafică a celor 3 axe pe care lucrează accelerometrul

1.3 Descrierea proiectului

În acest proiect, am creat o aplicație pentru Android ce presupune existența unui obiect, în cazul de față o bilă ce poate fi deplasată pe ecran în funcție de înclinația și orientarea telefonului. De exemplu, înclinarea telefonului într-o anumită direcție va cauza accelerarea bilei în direcția respectivă, urmând ca apoi, o înclinare în direcția opusă să cauzeze o decelerare treptată a bilei, iar apoi deplasarea sa accelerată în noua direcție.

1.4 Materiale necesare și folosite

1. Scrierea codului - Android Studio - 2021.2.1.15 - 64bit;
2. Emulator pentru testarea aplicației - Nexux 6 API 30;

3. Telefon fizic pentru instalarea și prezentarea aplicației - Xiaomi Poco X3 Pro;

1.5 Specificațiile telefonului folosit pentru instalare și testare

1. CPU - Qualcomm Snapdragon 860 - Octa-core - 2.96Ghz;
2. RAM - 8GB;
3. Versiune Android - 12 SKQ1.211006.001;

2 Prezentarea codului

2.1 Elementele globale

Pentru a putea avea o privire de ansamblu a aplicației, în cele ce urmează voi lucra cu variabile globale ce reprezintă diferite concepte precum:

1. Poziția bilei pe axele Ox și Oy așa cum se observă în imaginea de mai sus;
2. Accelerația bilei pe cele două axe;
3. Viteza bilei pe cele două axe;
4. Pozițiile maxime și minime posibile pe cele două axe deoarece nu dorim părăsirea ecranului de către bilă;
5. Bila ca obiect în sine;
6. Un manager de senzori ca instanță a clasei SensorManager pe care o pune la dispoziție Android Studio;

```
private float xPos, xAccel, xVel = 0.0f;
private float yPos, yAccel, yVel = 0.0f;
private float xMax, yMax;
private Bitmap ball;
private SensorManager sensorManager;
```

Fig. 2. Datele globale folosite de către aplicație

2.2 Desfășurarea programului

2.2.1 Crearea aplicației Ca idee de început, deoarece folosim accelerometrul, ar fi o idee foarte bună să obligăm telefonul să mențină aplicația doar în format de portret, deoarece, dacă înclinăm prea mult ecranul este posibil ca telefonul să schimbe orientarea și nu ne dorim acest lucru.

În cele ce urmează am creat bila noastră care este de fapt o imagine preluată de pe internet căreia i-am eliminat fundalul pentru o estetică mai bună și am setat vizualizarea ecranului să fie această bilă. De asemenea, tot în această etapă am preluat dimensiunile maxime ale ecranului deoarece nu dorim ca bila să părăsească ecranul. Ulterior am implementat practic un ecran circular, adică, de exemplu, dacă bila părăsește ecranul într-o anumită direcție, aceasta se va întoarce pe ecran prin latura opusă.

În final am instanțiat managerul de senzori ca o referință la instanța similară ce se regăsește pe sistemul telefonului.

```
@SuppressLint("SourceLockedOrientationActivity")
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
    BallView ballView = new BallView( context: this);
    setContentView(ballView);

    Point size = new Point();
    Display display = getWindowManager().getDefaultDisplay();
    display.getSize(size);

    this.xMax = (float) size.x - 100;
    this.yMax = (float) size.y - 100;

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

Fig. 3. Crearea efectivă a aplicației

2.2.2 Inițializarea și oprirea aplicației Acum este momentul în care trebuie să îi spunem aplicației că aceasta trebuie să asculte de accelerometru din momentul în care este pornită până în momentul în care o închidem.

Aici ne bazăm pe faptul că aceste metode sunt întotdeauna apelate înainte ca jocul să fie vizibil pentru utilizator și după ce aplicația este închisă definitiv.

În altă ordine de idei, tot ceea ce facem în partea de start este să înregistrăm accelerometrul ca senzor de care ar trebui să asculte aplicația la un anumit refresh rate, mai exact constanta ”SENSOR_DELAY_GAME”, iar în partea de oprire să oprim acest comportament.

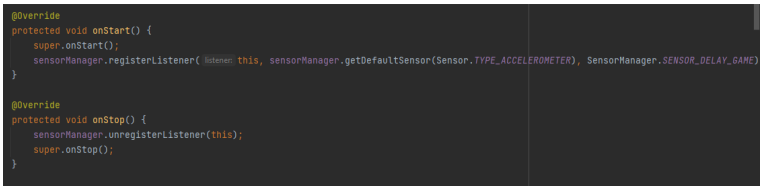

Fig. 4. Acțiunile din onStart() și onStop()

2.2.3 Preluarea datelor de la accelerometru Pentru a prelua datele de la accelerometru suntem nevoiți să implementăm metoda „onSensorChanged" din interfața „SensorEventListener2".

Pentru a face acest lucru, conform documentației din Android, datele de care avem nevoie, adică informația pe care o primim de la accelerometru (accelerația) se regăsește în „sensorEvent.values[i]" unde i poate primi o valoare cuprinsă între 0 și 2, fiecare dintre valori fiind corespunzătoare unei axe;

Aplicația noastră se desfășoară doar în două dimensiuni așa încât o dată pe cadru, preluăm accelerația de la accelerometru și o plasăm în cele două variabile globale destinate pentru acest lucru. De asemenea suntem nevoiți să actualizăm poziția bilei.

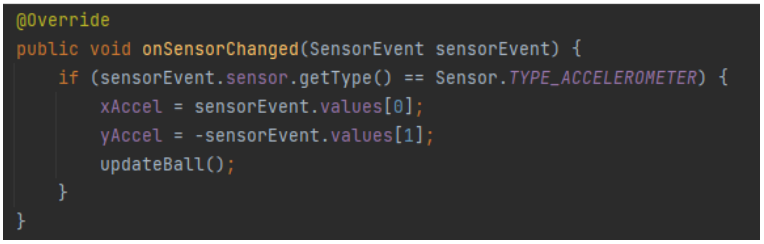

Fig. 5. Preluarea datelor de la accelerometru

2.2.4 Modalitatea de actualizarea a poziției bilei Actualizarea efectivă a poziției bilei se face în metoda „updateBall" unde ne folosim de acceleratiile preluate de la accelerometru pentru a calcula viteza bilei conform legilor fizicii. Practic, viteza pe o anumită axă este reprezentată de accelerația pe acea axă înmulțită cu durata unui frame.

De asemenea, aici este locul în care realizăm o hartă circulară pentru bila noastră în ideea în care fixăm o poziție minimă și una maximă pentru fiecare

dintre axe, iar în momentul depășirii acestora, bila se va întoarece în ecran prin partea opusă.

```
private void updateBall() {
    float frameTime = 0.666f;
    xVel += (xAccel * frameTime);
    yVel += (yAccel * frameTime);
    float xS = (xVel / 2) * frameTime;
    float yS = (yVel / 2) * frameTime;
    xPos -= xS;
    yPos -= yS;
    if (xPos > xMax) {
        xPos = 0;
    } else if (xPos <= 0) {
        xPos = xMax;
    }

    if (yPos > yMax) {
        yPos = 0;
    } else if (yPos <= 0) {
        yPos = yMax;
    }
}
```

Fig. 6. Actualizarea efectivă a poziției bilei

2.2.5 Afișarea bilei pe ecran Având în vedere faptul că întreaga aplicație se referă la vizualizarea unei bile o putem pune pur și simplu pe ecran având grijă ca dimensiunile hărții pe care se află să fie scalate la dimensiunea fiecărui ecran de telefon pe care această aplicație va fi instalată.

În acest sens, suntem nevoiți să suprascriem o metodă ce se numește „onDraw" în așa fel încât aceasta să fie apelată de fiecare dată când poziția bilei se schimbă.

```
private class BallView extends View {
    public BallView(Context context) {
        super(context);
        Bitmap ballSrc = BitmapFactory.decodeResource(getResources(), R.drawable.ball);
        final int dstWidth = 100;
        final int dstHeight = 100;
        ball = Bitmap.createScaledBitmap(ballSrc, dstWidth, dstHeight, filter: true);
    }

    @Override
    protected void onDraw(Canvas canvas) {
        canvas.drawBitmap(ball, xPos, yPos, paint: null);
        invalidate();
    }
}
```

Fig. 7. Desenarea și afișarea bilei pe hartă/ecran

3 Concluzii

În acest proiect am învățat cum să accesez accelerometrul de pe un dispozitiv Android și cum să creez un joc în Android Studio folosind datele pe care el le furnizează. Practic, am construit un joc ce surprinde mișcarea unei bile ce este strâns legată de către mișcarea telefonului și implict a accelerometrului.

4 Componența arhivei

1. Documentația proiectului
2. Arhiva cu resursele utilizate, inclusiv codul din Android Studio
3. Fișierul APK pentru instalarea jocului pe orice dispozitiv se dorește

5 Referințe

1. https://developer.android.com/studio
2. https://www.youtube.com/watch?v=pkT7DU1Yo9Q
3. https://code.tutsplus.com/tutorials/using-the-accelerometer-on-android–mobile-22125
4. https://gamedevacademy.org/android-sensors-game-tutorial/
5. https://examples.javacodegeeks.com/android/core/hardware/sensor/android-accelerometer-example/