Министерство науки и высшего образования Российской Федерации

Севастопольский государственный университет

Кафедра ИС

Отчет

По дисциплине: «Разработка мобильных приложений»

Лабораторная работа №4

«Исследование датчиков в ОС Android»

Выполнил

ст. гр. ИС/б-17-2о

Волобуев Ю.С.

Проверил:

Шишкевич В.Е.

Севастополь

2020

1 ЦЕЛЬ

Исследование процесса разработки практического мобильного приложения с использованием датчиков.

2 ВАРИАНТ ЗАДАНИЯ

Вариант 9:

|  |  |
| --- | --- |
| № варианта | Задание |
| 9 | Создать приложение, которое при вращении смартфона против часовой стрелки делает экран желтого цвета, а при вращении по часовой стрелке зеленым |

3 ХОД РАБОТЫ

1. Листинг программы:

package com.example.sensorapplication;  
  
import androidx.appcompat.app.AppCompatActivity;  
  
import android.content.Context;  
import android.hardware.Sensor;  
import android.hardware.SensorManager;  
import android.hardware.SensorEvent;  
import android.hardware.SensorEventListener;  
import android.os.Bundle;  
import android.view.Display;  
import android.view.Surface;  
import android.view.WindowManager;  
import android.widget.RelativeLayout;  
import android.widget.TextView;  
  
import java.util.ArrayList;  
import java.util.Timer;  
import java.util.TimerTask;  
  
public class MainActivity extends AppCompatActivity {  
 SensorManager sensorManager;  
 TextView textView;  
 TextView textView1;  
 RelativeLayout relativeLayout;  
 Sensor sensorAccel;  
 Sensor sensorMagnet;  
  
 ArrayList<Float> y\_Axis = new ArrayList<>();  
 StringBuilder sb = new StringBuilder();  
 Timer timer;  
 int rotation;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
  
 sensorManager = (SensorManager) getSystemService(*SENSOR\_SERVICE*);  
 sensorAccel = sensorManager.getDefaultSensor(Sensor.*TYPE\_ACCELEROMETER*);  
 sensorMagnet = sensorManager.getDefaultSensor(Sensor.*TYPE\_MAGNETIC\_FIELD*);  
 textView = (TextView) findViewById(R.id.*Turn*);  
 textView1 = (TextView) findViewById(R.id.*RotateWord*);  
 relativeLayout = (RelativeLayout) findViewById(R.id.*RelativeLayout01*);  
 }  
  
 @Override  
 protected void onResume() {  
 super.onResume();  
 sensorManager.registerListener(listener, sensorAccel, SensorManager.*SENSOR\_DELAY\_NORMAL*);  
 sensorManager.registerListener(listener, sensorMagnet, SensorManager.*SENSOR\_DELAY\_NORMAL*);  
  
 timer = new Timer();  
 TimerTask task = new TimerTask() {  
 @Override  
 public void run() {  
 runOnUiThread(new Runnable() {  
 @Override  
 public void run() {  
 getDeviceOrientation();  
 getActualDeviceOrientation();  
 showInfo();  
 for(Float yAxis : y\_Axis){  
 int index = y\_Axis.indexOf(yAxis);  
 if(index > 1 && yAxis.compareTo(y\_Axis.get(index-1)) == 1){  
 textView1.setText(getResources().getText(R.string.*turn\_clockwise*));  
 relativeLayout.setBackgroundColor(getResources().getColor(R.color.*turn\_clockwise*));  
 }  
 else if (index > 1 && yAxis.compareTo(y\_Axis.get(index-1)) == -1){  
 textView1.setText(getResources().getText(R.string.*turn\_counterclockwise*));  
 relativeLayout.setBackgroundColor(getResources().getColor(R.color.*turn\_counterclockwise*));  
 }  
 }  
 }  
 });  
 }  
 };  
 timer.schedule(task, 0, 300);  
  
 WindowManager windowManager = ((WindowManager) getSystemService(Context.*WINDOW\_SERVICE*));  
 Display display = windowManager.getDefaultDisplay();  
 rotation = display.getRotation();  
 }  
  
 @Override  
 protected void onPause() {  
 super.onPause();  
 sensorManager.unregisterListener(listener);  
 timer.cancel();  
 }  
  
 String format(float values[]) {  
 return String.*format*("%1$.1f\t\t%2$.1f\t\t%3$.1f", values[0], values[1], values[2]);  
 }  
  
 void showInfo() {  
 sb.setLength(0);  
 sb.append("Orientation : " + format(valuesResult))  
 .append("\nOrientation 2: " + format(valuesResult2))  
 ;  
 textView.setText(sb);  
 }  
  
 float[] r = new float[9];  
  
 void getDeviceOrientation() {  
 SensorManager.*getRotationMatrix*(r, null, valuesAccel, valuesMagnet);  
 SensorManager.*getOrientation*(r, valuesResult);  
  
 valuesResult[0] = (float) Math.*toDegrees*(valuesResult[0]);  
 valuesResult[1] = (float) Math.*toDegrees*(valuesResult[1]);  
 valuesResult[2] = (float) Math.*toDegrees*(valuesResult[2]);  
  
 y\_Axis.add(valuesResult[2]);  
  
 return;  
 }  
 float[] inR = new float[9];  
 float[] outR = new float[9];  
  
 void getActualDeviceOrientation() {  
 SensorManager.*getRotationMatrix*(inR, null, valuesAccel, valuesMagnet);  
 int x\_axis = SensorManager.*AXIS\_X*;  
 int y\_axis = SensorManager.*AXIS\_Y*;  
 switch (rotation) {  
 case (Surface.*ROTATION\_0*): break;  
 case (Surface.*ROTATION\_90*):  
 x\_axis = SensorManager.*AXIS\_Y*;  
 y\_axis = SensorManager.*AXIS\_MINUS\_X*;  
 break;  
 case (Surface.*ROTATION\_180*):  
 y\_axis = SensorManager.*AXIS\_MINUS\_Y*;  
 break;  
 case (Surface.*ROTATION\_270*):  
 x\_axis = SensorManager.*AXIS\_MINUS\_Y*;  
 y\_axis = SensorManager.*AXIS\_X*;  
 break;  
 default: break;  
 }  
 SensorManager.*remapCoordinateSystem*(inR, x\_axis, y\_axis, outR);  
 SensorManager.*getOrientation*(outR, valuesResult2);  
 valuesResult2[0] = (float) Math.*toDegrees*(valuesResult2[0]);  
 valuesResult2[1] = (float) Math.*toDegrees*(valuesResult2[1]);  
 valuesResult2[2] = (float) Math.*toDegrees*(valuesResult2[2]);  
 return;  
 }  
  
 float[] valuesAccel = new float[3];  
 float[] valuesMagnet = new float[3];  
 float[] valuesResult = new float[3];  
 float[] valuesResult2 = new float[3];  
  
  
 SensorEventListener listener = new SensorEventListener() {  
  
 @Override  
 public void onAccuracyChanged(Sensor sensor, int accuracy) {  
 }  
  
 @Override  
 public void onSensorChanged(SensorEvent event) {  
 switch (event.sensor.getType()) {  
 case Sensor.*TYPE\_ACCELEROMETER*:  
 for (int i=0; i < 3; i++){  
 valuesAccel[i] = event.values[i];  
 }  
 break;  
 case Sensor.*TYPE\_MAGNETIC\_FIELD*:  
 for (int i=0; i < 3; i++){  
 valuesMagnet[i] = event.values[i];  
 }  
 break;  
 }  
 }  
 };  
}

В методе onCreate()  мы получаем сенсоры ускорения (TYPE\_ACCELEROMETER) и магнитного поля ([TYPE\_MAGNETIC\_FIELD](http://developer.android.com/reference/android/hardware/Sensor.html#TYPE_MAGNETIC_FIELD)).

В onResume() вешаем слушателя и запускаем таймер, который каждые 300 мсек будет определять ориентацию девайса в пространстве и выводить эту информацию на экран. В переменную rotation получаем значение текущей ориентации экрана. Это нам понадобиться для корректного определения ориентации девайса.

В onPause() отключаем слушателя и таймер.

Метод getDeviceOrientation() определяет текущую ориентацию девайса в пространстве без учета поворота экрана. Для этого сначала вызывается метод [getRotationMatrix](http://developer.android.com/reference/android/hardware/SensorManager.html" \l "getRotationMatrix(float[],%20float[],%20float[],%20float[])" \t "_blank), который берет данные ускорения и магнитного поля и формирует из них матрицу данных в переменную r. Далее метод [getOrientation](http://developer.android.com/reference/android/hardware/SensorManager.html" \l "getOrientation(float[],%20float[])" \t "_blank) из этой матрицы позволяет получить массив значений (в радианах) поворота трех осей.

Слушатель listener получает данные ускорения и магнитного поля и запоминает их в массивы valuesAccel и valuesMagnet.

1. Протестируем работу приложения (см. рисунок 1).

При повороте экрана по часовой стрелке или против часовой стрелки выводится соответствующее сообщение, а также оrientation: данные по ориентации в пространстве без учета ориентации экрана устройства.  
Orientation 2: данные по ориентации в пространстве с учетом ориентации экрана устройства. Первое число – значения оси z, затем x и y.

Изображение выглядит как снимок экрана, птица, цветок

Автоматически созданное описание Изображение выглядит как снимок экрана, цветок, птица

Автоматически созданное описание

Рисунок 1 – Скриншоты работы приложения

ВЫВОДЫ

В ходе данной лабораторной работы был исследован процесс разработки практического мобильного приложения с использованием датчиков. Были изучены особенности работы с акселерометром, датчиком магнитного поля и ориентацией.