Лабораторная работа № 8. Разработка мобильных приложений с функциями распознавания жестов

Цели лабораторной работы:

- разработать простейшие приложения для демонстрации распознавания стандартных жестов;
- разработать приложения, распознающие жесты, вводимые пользователями.

Задачи лабораторной работы:

- рассмотреть распознавание всех поддерживаемых жестов;
- рассмотреть распознавание только части поддерживаемых жестов;
- создать набор жестов;
- использовать созданные жесты в приложении.

Table of Contents

Критерии оценивания

4 — приложения на основе примеров из лабораторной работы и ответы на контрольные вопросы;

5-6 — приложения на основе примеров из лабораторной работы, упражнение №1-2 для самостоятельной работы и ответы на контрольные вопросы. **Генерация кода на языке Kotlin из java-проекта не разрешается**;

7-8 — приложения на основе примеров из лабораторной работы, упражнения № 1-3 для самостоятельной работы и ответы на контрольные вопросы. **Генерация кода на языке Kotlin из java-проекта не разрешается**;

9 — приложения на основе примеров из лабораторной работы, все

упражнения для самостоятельной работы, дополнительный функционал, предложенный студентом самостоятельно, и ответы на контрольные вопросы. **Генерация кода на языке Kotlin из java-проекта не разрешается.**

ЗАДАНИЕ 8.1. Демонстрации распознавания стандартных жестов

Для работы со стандартными жестами Android предоставляет класс GestureDetector. Этот класс содержит два вложенных интерфейса-слушателя: OnGestureListener и OnDoubleTapListener, эти интерфейсы задают методы, отслеживающие стандартные жесты. А также GestureDetector содержит вложенный класс SimpleOnGestureListener, который содержит пустые реализации, возвращающие значение false, где это необходимо, всех методов интерфейсов: OnGestureListener и OnDoubleTapListener.

В лабораторной работе рассмотрим две возможности распознавания жестов:

- случай распознавания всех поддерживаемых жестов, для этого реализуем в классе активности оба интерфейса;
- случай распознавания только некоторого набора поддерживаемых жестов, для этого в классе активности объявим внутренний класс-наследник класса GestureDetector.SimpleOnGestureListener.

Получить больше информации о распознавании жестов можно по ссылке: http://developer.android.com/training/gestures/index.html

Упражнение 8.1.1 Распознавание всех поддерживаемых жестов

Разработать приложение, в котором продемонстрировать распознавание поддерживаемых стандартных жестов.

Рассмотрим использование стандартных жестов на примере. Приложение содержит одну активность, одно информационное поле для вывода информации о распознанном жесте. Приложение работает следующим образом: пользователь выполняет один из поддерживаемых сенсорных жестов, в информационном поле отображается информация о распознанном жесте.

1. Создадим простое приложение и добавим на форму TextView для вывода информации.
2. Настроим логику приложения. В java класс, соответствующий активности внесем следующие дополнения.

Класс активности должен реализовывать интерфейсы: GestureDetector.OnGestureListener и GestureDetector.OnDoubleTapListener, для этого в объявление класса добавим конструкцию:

```
implements GestureDetector.OnGestureListener,
           GestureDetector.OnDoubleTapListener
```

Нам понадобится экземпляр класса GestureDetectorCompat поэтому в качестве

поля класса активности объявим следующую переменную:

```
GestureDetectorCompat mDetector;
```

В методе onCreate() класса активности, создадим экземпляр класса GestureDetectorCompat и присвоим его переменной mDetector:

```
mDetector = new GestureDetectorCompat(this,this);
```

Одним из параметров конструктора является класс, который реализует интерфейс GestureDetector.OnGestureListener, в нашем случае использовано слово this, т. е. параметром является сам класс активности. Этот интерфейс уведомляет пользователей когда появляется определенное сенсорное событие.

В методе OnCreate() класса активности, следующая строка:

```
mDetector.setOnDoubleTapListener(this);
```

устанавливает слушатель событий, связанных с двойным касанием, это должен быть класс, реализующий интерфейс GestureDetector.OnDoubleTapListener. В нашем случае использовано слово this, т.е. слушателем будет опять сам класс активности.

1. Чтобы позволить вашему объекту GestureDetector получать события, необходимо переопределить метод onTouchEvent() для активности или элемента GUI. И передавать в экземпляр детектора все обнаруженные события.

```
public boolean onTouchEvent(MotionEvent event){
    this.mDetector.onTouchEvent(event);
    // Be sure to call the superclass implementation
    return super.onTouchEvent(event);
}
```

2. После проведенной подготовки пришло время реализовать все методы, объявленные в интерфейсах, отвечающих за прослушивание сенсорных событий.

Методы интерфейса GestureDetector.OnGestureListener:

onDown()	- отслеживает появление касания, т. е. палец прижат к экрану;
onFling()	- отслеживает появление жеста смахивания;

onLongPress()	- отслеживает удерживание пальца прижатым к экрану длительное время;
onScroll()	- отслеживает появление жеста прокрутки (пролистывания);
onShowPress()	- отслеживает, что произошло событие касания и больше никаких событий не происходит короткое время;
onSingleTapUp()	- отслеживает появление жеста одиночного нажатия (клик).

Методы интерфейса GestureDetector.OnDoubleTapListener:

onDoubleTap()	- отслеживает появление жеста двойного нажатия ("двойной клик");
onDoubleTapEvent()	- отслеживает появление события во время выполнения жеста двойного нажатия, включая касание, перемещение, подъем пальца.
onSingleTapConfirmed()	- отслеживает появление жеста одиночного нажатия (клик).

В листинге 8.1 представлен код приложения, в котором распознаются все поддерживаемые жесты, информация о появившемся и распознанном жесте выдается в информационное поле (TextView).

Упражнение 8.1.2 Распознавание только части поддерживаемых жестов.

Разработать приложение, в котором продемонстрировать распознавание только некоторой части поддерживаемых жестов, например смахивания (fling), перетаскивания (Dragging), масштабирования (scaling) и обработки жеста Multi-Touch.

Разработаем приложение, в котором продемонстрируем распознавание только некоторой части поддерживаемых жестов по выбору программиста.

Мы рассмотрим распознавание жеста смахивания (fling). Приложение содержит одну активность, одно информационное поле для вывода информации о распознанном жесте. Приложение работает следующим образом: пользователь выполняет один из поддерживаемых сенсорных жестов, в информационном поле отображается информация о распознанном жесте.

1. Создадим простое приложение и добавим на форму TextView для вывода информации.
2. Настроим логику приложения. В java класс, соответствующий активности внесем следующие дополнения.

Нам понадобится экземпляр класса GestureDetectorCompat поэтому в качестве поля класса активности объявим следующую переменную:

```
GestureDetectorCompat mDetector;
```

В методе onCreate() класса активности, создадим экземпляр класса GestureDetectorCompat и присвоим его переменной mDetector:

```
mDetector=new GestureDetectorCompat(this, new MyGestListener());
```

в конструкторе аргументом, отвечающим за отслеживание сенсорных событий, служит экземпляр класса MyGestListener() - внутренний класс, который является наследником класса GestureDetector.SimpleOnGestureListener.

Имеет смысл немного рассмотреть класс GestureDetector.SimpleOnGestureListener. Этот класс реализует интерфейсы GestureDetector.OnGestureListener и GestureDetector.OnDoubleTapListener, все методы заявленные в интерфейсах в этом классе имеют пустую реализацию и те, которые должны возвращать значение, возвращают false. Поэтому для распознавания какого-то события или некоторого подмножества событий достаточно написать реализацию соответствующих методов, в классе наследнике.

В листинге 8.2 представлен код приложения, в котором распознается только жест смахивания, т. е. реализован метод onFling(), информация о появившемся и распознанном жесте выдается в информационное поле (TextView). В качестве слушателя используется экземпляр класса MyGestListener(), являющийся наследником класса GestureDetector.SimpleOnGestureListener.

Листинг 8.1. Распознавание поддерживаемых жестов с помощью реализации интерфейсов

```
package com.example.lab5_1_gestall;

import android.os.Bundle;
import android.app.Activity;
import android.support.v4.view.GestureDetectorCompat;
import android.view.*;
import android.widget.*;

public class MainActivity extends Activity
       implements GestureDetector.OnGestureListener,
                  GestureDetector.OnDoubleTapListener
{
  TextView tvOutput;
  GestureDetectorCompat mDetector;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    tvOutput = (TextView)findViewById(R.id.textView1);

    mDetector = new GestureDetectorCompat(this,this);
    mDetector.setOnDoubleTapListener(this);
```

```
  }

  public boolean onTouchEvent(MotionEvent event){
    this.mDetector.onTouchEvent(event);
    // Be sure to call the superclass implementation
    return super.onTouchEvent(event);
      }

  @Override
  public boolean onDown(MotionEvent event) {
      tvOutput.setText("onDown: " + event.toString());
      return false;
  }

  @Override
  public boolean onFling(MotionEvent event1, MotionEvent event2,
                 float velocityX, float velocityY) {
    tvOutput.setText("onFling: " + event1.toString()+event2.toString());

      return true;
  }

  @Override
  public void onLongPress(MotionEvent event) {
     tvOutput.setText("onLongPress: " + event.toString());
  }

  @Override
  public boolean onScroll(MotionEvent e1, MotionEvent e2, float distanceX,
                    float distanceY) {
     tvOutput.setText("onScroll: " + e1.toString()+e2.toString());
       return true;
  }

  @Override
  public void onShowPress(MotionEvent event) {
     tvOutput.setText("onShowPress: " + event.toString());
  }

  @Override
  public boolean onSingleTapUp(MotionEvent event) {
    tvOutput.setText("onSingleTapUp: " + event.toString());
      return true;
  }

  @Override
  public boolean onDoubleTap(MotionEvent event) {
    tvOutput.setText("onDoubleTap: " + event.toString());
      return true;
  }

  @Override
  public boolean onDoubleTapEvent(MotionEvent event) {
    tvOutput.setText("onDoubleTapEvent: " + event.toString());
      return true;
  }

  @Override
  public boolean onSingleTapConfirmed(MotionEvent event) {
     tvOutput.setText("onSingleTapConfirmed: " + event.toString());
      return true;
  }
}
```

Листинг 8.2. Распознавание жестов с использованием класса GestureDetector.SimpleOnGestureListener

```
package com.example.lab5_1_gestsubset;

import android.os.Bundle;
import android.app.Activity;
import android.support.v4.view.*;
import android.view.*;
import android.widget.*;

public class SubsetGestActivity extends Activity {

private GestureDetectorCompat mDetector;
private TextView tvOut;

@Override
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_subset_gest);
  mDetector = new GestureDetectorCompat(this, new MyGestListener());
  tvOut = (TextView)findViewById(R.id.textView1);
}

@Override
public boolean onTouchEvent(MotionEvent event){
this.mDetector.onTouchEvent(event);
return super.onTouchEvent(event);
}

class MyGestListener extends GestureDetector.SimpleOnGestureListener {

@Override
public boolean onFling(MotionEvent event1, MotionEvent event2,
                  float velocityX, float velocityY) {
tvOut.setText("onFling: " + event1.toString()+event2.toString());
   return true;
  }
}

}
```

ЗАДАНИЕ 8.2. Принципы работы с жестами вводимыми пользователями

Начиная с версии 1.6, Android предоставляет API для работы с жестами, который располагается в пакете android.gesture и позволяет сохранять, загружать, создавать и распознавать жесты.

В данной части лабораторной работы 8 рассмотрим процесс создания набора жестов, для создания жестов будем использовать приложение Gesture Builder. Приложение Gesture Builder по умолчанию связано с профилем эмулятора AVD для некоторых версий SDK. Однако он не предустановлен на большинстве физических устройств Android. Если утилита предустановлена, она будет указана вместе с другими приложениями, установленными на устройстве или экземпляре AVD. В случае, если он не установлен, пожалуйста, установите его на эмуляторе из Google Apps.

Реализовать приложение, рассматриваемое в примере, в котором предполагается распознавание и использование созданных жестов.

Упражнение 8.2.1 Создание набора жестов

Просмотрите видео с демонстрацией использования кастомных жестов https://youtu.be/7YKuFkdcdhY.

Для начала создадим новое приложение.

Далее запустим эмулятор и используем приложение Gesture Builder, например https://play.google.com/store/apps/details?id=migueldp.runeforge, для создания жестов "1", "2" и "S". Если не получится установить приложение на эмуляторе, то установить на телефон и создать жесты.

Жест всегда связан с именем, но имя не обязательно должно быть уникальным, в действительности, для повышения точности в распознавании жеста рекомендуется сохранять несколько жестов с одним и тем же именем.

На рис. 8.1 можно увидеть приложение Gesture Builder в работе, чтобы добавить жест, необходимо нажать на кнопку **Add gesture**, в свободном пространстве изобразить жест (обычно он рисуется желтым цветом), в поле ввода задать имя жеста. Результат последовательного добавления жестов можно увидеть на рис. 8.2.

Жесты сохраняются на SD карте эмулятора, чтобы использовать их в приложении необходимо импортировать файл жестов в проект.

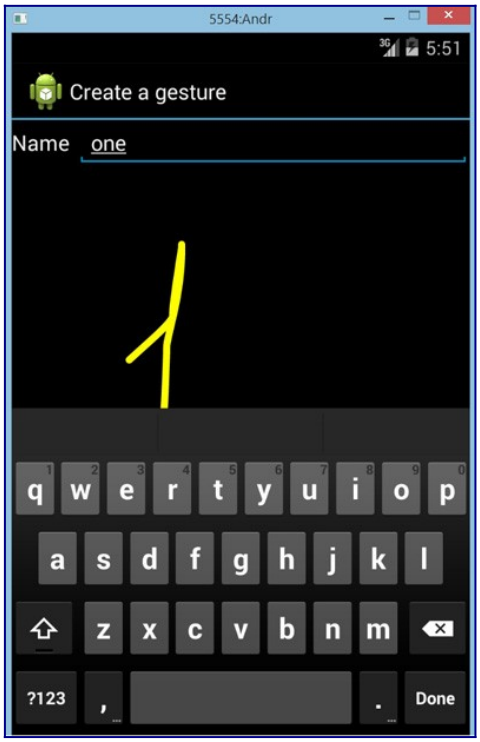

Рис. 8.1. Создание жестов с помощью приложения Gesture Builder

Рис. 8.2. Набор жестов, созданных в приложении Gesture Builder

Комментарий: Gesture Builder может сообщить, что ему некуда сохранять жесты, в этом случае необходимо запустить эмулятор с образом SD карты.

Сначала образ нужно создать с помощью утилиты **mksdcard** (расположена в папке **<AndrSDK>/sdk/tools**, где **<AndrSDK>** - путь, куда установлен Android SDK). В командной строке напишем:

```
mksdcard -l mySdCard 64M gesture.img
```

Следующим шагом будет создать и запустить эмулятор с образом gesture.img, для этого зайдем в папку **<AndrSDK>/sdk/tools** и выполним команду:

```
emulator -avd nameEmulator -sdcard gestures.img
```

nameEmulator - имя, которое присвоено эмулятору при создании.

Интересная статья на эту тему по ссылке: http://habrahabr.ru/post/120016/

Каждый раз, когда мы создаем или редактируем жесты с помощью Gesture Builder, создается файл gestures на SD карте эмулятора. Необходимо импортировать этот файл в директорию **res/raw/**, созданного проекта, в котором планируем использовать жесты.

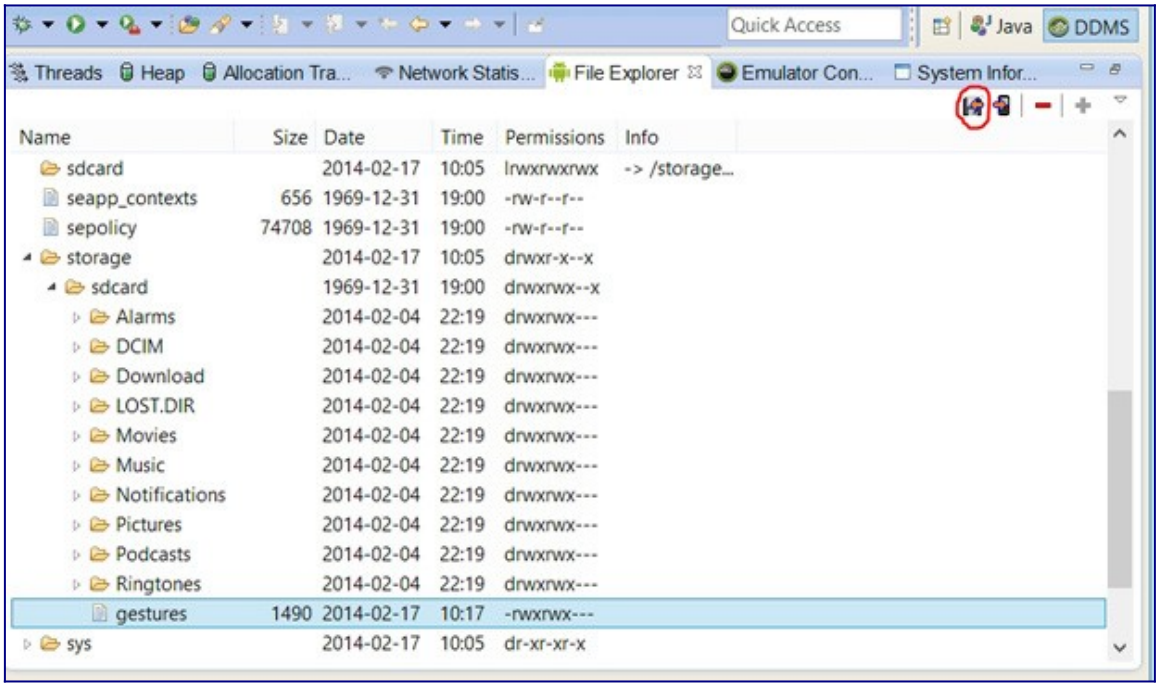

Рис. 8.3. Расположение файла gestures

Самый простой способ импортировать жесты в проект заключается использовании вкладки **File Explorer** в компоновке (perspective) DDMS. (Если компоновки DDMS нет найти ее можно следующим образом: **Window->Open Perspective->Other...->DDMS**. Если вкладки **File Explorer** нет, можно добавить: **Window-> Show View-> File Explorer**). На вкладке **File Explorer** найти директорию **sdcard/** (в нашем случае оказалась директория **storage/sdcard/**, имеет смысл при создании жестов обратить внимание в какую директорию Gesture Builder их сохраняет). На рис. 8.3 показана вкладка **File Explorer** в компоновке DDMS.

Чтобы скопировать файл жестов с эмулятора в проект, необходимо выбрать его и нажать кнопку "**Pull a file from the device**", выделенную на рис. 8.3 красным подобием окружности. Откроется диалог с предложением выбрать папку, в которую необходимо скопировать жесты, здесь надо найти папку проекта, в ней папку **res/raw/** (если папки **raw/** нет, ее необходимо создать) и нажать кнопку **Сохранить**. Теперь жесты есть в нашем проекте и их можно использовать.

Упражнение 8.2.2 Использование созданных жестов в приложении

Для распознавания жестов необходимо добавить элемент GestureOverlayView в XML файл активности. И этот файл может выглядеть, например, как показано на рис. 8.4:

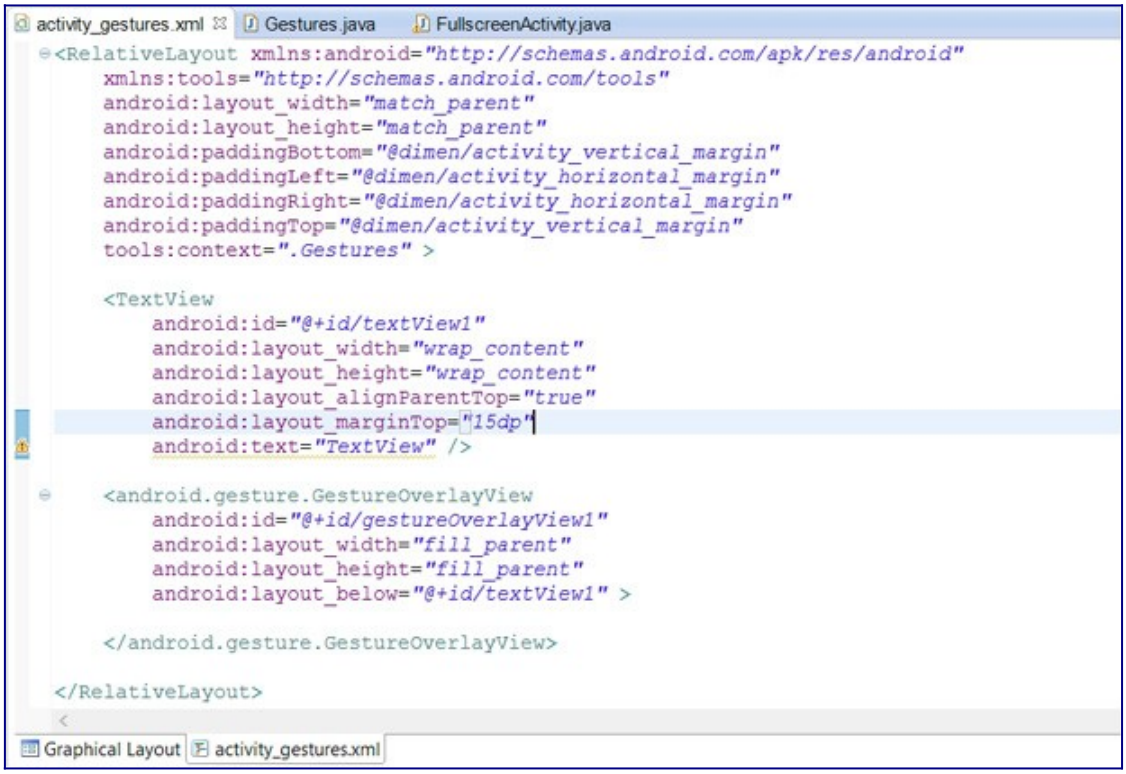

Рис. 8.4. XML файл активности приложения, элемент GestureOverlayView обычный компонент интерфейса пользователя

Можно добавить элемент GestureOverlayView поверх всех компонентов, как прозрачный слой, в этом случае XML файл активности может выглядеть так, как показано на рис. 8.5.

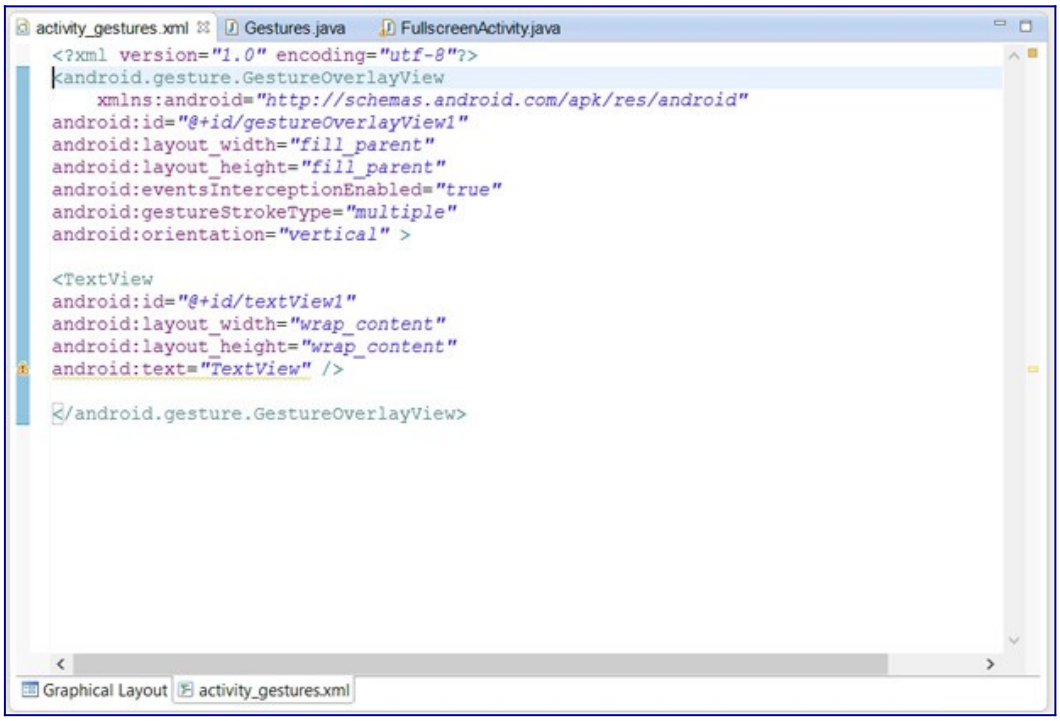

Рис. 8.5. XML файл активности приложения, элемент GestureOverlayView поверх всех компонентов интерфейса пользователя

Далее необходимо обработать ввод жеста пользователя, сравнить с загруженными жестами, и либо определить жест, либо сообщить пользователю, что такого жеста нет. Теперь вся работа будет выполняться в java файле, описывающем

главную (и единственную) активность приложения. Внесем в этот класс следующие дополнения:

- Класс активности должен реализовывать интерфейс OnGesturePerformedListener, для этого в объявление класса добавим конструкцию:

```
implements OnGesturePerformedListener;
```

Нам понадобятся экземпляры классов GestureLibrary и GestureOverlayView, поэтому в качестве полей класса активности объявим следующие переменные:

```
GestureLibrary gLib;
GestureOverlayView gestures;
```

В методе onCreate() выполним следующие действия:

```
gLib = GestureLibraries.fromRawResource(this, R.raw.gestures);
if (!gLib.load()) {
finish();
}
```

В первой строке выполнена инициализация переменной gLib жестами, загруженными из файла gestures папки **res/raw/**.

Оператор if выполняет проверку загружены ли жесты, если нет, выполняется выход из приложения.

Добавим в метод onCreate() еще две строчки:

```
gestures = (GestureOverlayView)  findViewById(R.id.gestureOverlayView1);
gestures.addOnGesturePerformedListener(this);
```

Для инициализации переменной gesture и подключения к ней слушателя событий появления жеста.

И наконец напишем реализацию метода OnGesturePerformed(), который и будет вызываться при появлении события, соответствующего какому-либо жесту.

```
public void onGesturePerformed(GestureOverlayView overlay, Gesture
gesture) {
    //Создаёт ArrayList с загруженными из gestures жестами
        ArrayList<Prediction> predictions = gLib.recognize(gesture);
        if (predictions.size() > 0) {
          //если загружен хотябы один жест из gestures
            Prediction prediction = predictions.get(0);
            if (prediction.score > 1.0) {
                if (prediction.name.equals("one"))
                tvOut.setText("1");
```

```
                    else if (prediction.name.equals("stop"))
                      tvOut.setText("stop");
                    else if (prediction.name.equals("two"))
                      tvOut.setText("2");
                  }else{
                    tvOut.setText("Жест неизвестен");
                  }
              }
          }
```

В приложении всего лишь распознаются жесты и в информационное поле выводится информация о том, что за жест был использован. В листинге 8.3 представлен возможный код приложения.

Листинг 8.3. Распознавание жестов загруженных в файл res/raw/gestures

```
package com.example.lab5_2_gestures;

import java.util.ArrayList;

import android.os.Bundle;
import android.app.Activity;
import android.gesture.Gesture;
import android.gesture.GestureLibraries;
import android.gesture.GestureLibrary;
import android.gesture.GestureOverlayView;
import android.gesture.GestureOverlayView.OnGesturePerformedListener;
import android.gesture.Prediction;
import android.view.Menu;
import android.widget.TextView;

public class Gestures extends Activity implements OnGesturePerformedListener{

  GestureLibrary gLib;
  GestureOverlayView gestures;
  TextView tvOut;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_gestures);
    tvOut=(TextView)findViewById(R.id.textView1);
        //Загрузка жестов (gestures) из res/raw/gestures
        gLib = GestureLibraries.fromRawResource(this, R.raw.gestures);
        if (!gLib.load()) {
          //Если жесты не загружены, то выход из приложения
            finish();
        }

       gestures = (GestureOverlayView) findViewById(R.id.gestureOverlayView1);
       gestures.addOnGesturePerformedListener(this);
  }

  @Override
  public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.gestures, menu);
    return true;
  }
```

```
  public void onGesturePerformed(GestureOverlayView overlay, Gesture gesture) {
    //Создаёт ArrayList с загруженными из gestures жестами
        ArrayList<Prediction> predictions = gLib.recognize(gesture);
        if (predictions.size() > 0) {
          //если загружен хотя бы один жест из gestures
            Prediction prediction = predictions.get(0);
            if (prediction.score > 1.0) {
                if (prediction.name.equals("one"))
                tvOut.setText("1");
                else if (prediction.name.equals("stop"))
                  tvOut.setText("stop");
                else if (prediction.name.equals("two"))
                  tvOut.setText("2");
            }else{
              tvOut.setText("Жест неизвестен");
            }
        }
    }

}
```

ЗАДАНИЕ 8.3 Самостоятельна работа

УПРАЖНЕНИЕ 1

Разработать приложение, использующее поддерживаемые жесты. Продемонстрировать использование всех жестов или их части.

УПРАЖНЕНИЕ 2

В качестве практического задания реализовать ввод чисел жестами в приложении "Угадайка", разработанном в лабораторной работе № 6. Создать жесты "0", "1", "2", "3", "4", "5", "6", "7", "8", "9" для ввода цифр и предложить свой жест для остановки ввода числа. В приложение добавить распознавание жестов, преобразование их в число и сравнение полученного числа с загаданным.

УПРАЖНЕНИЕ 3

Внести изменения в приложение калькулятор и добавить жестовый ввод чисел и операций.

УПРАЖНЕНИЕ 4

Разработать блокнот на языке Kotlin для заметок с рукописным вводом текста.

Контрольные вопросы

1. Вызов какого метод инициируется при появлении сенсорного события? При каком условии возможна обработка жеста (как должен быть реализован метод)?
2. Какой класс позволяет распознавать стандартные жесты без обработки отдельных сенсорных событий?
3. Перечислите методы, отвечающие за прослушивание сенсорных событий.
4. С помощью какого приложения можно создавать свои жесты и добавлять их в виде бинарного ресурса в свое приложение?

5. Какой элемент требуется добавить в XML-файл активности для распознавания созданных (кастомных) жестов? Какие способы его добавления?
6. Какой интерфейс должен реализовывать класс активности при обработке созданных (кастомных) жестов?