Android Common Gestures

Android Gestures. This tutorial describes Gesture in android and how to use Gestures.

This tutorial is based on Android studio 2.2

What is Gesture:

The term “gesture” is used to define a contiguous sequence of interactions between the touch screen and the use. Android provides special types of touch screen events such as single tap, double tap, scrolls, long presses, show on and On Down. These are all known as gestures. We all agree that mobile technology has become a big part of the technology era. One of the reasons for the success of mobile is the ability to interact with users through gestures and accessibility features.

How to use Gesture:

We can use it by using some libraries and functions. Android provides GestureDetector class to receive motion events and tell us that these events correspond to gestures or not. GestureDetector is powerful for these standard user interactions and is easy to use it, we should have object of GestureDetector. The simplicity of this class lays in overriding the methods you need from the GestureListener and implementing the gesture functionality required without having to manually determine what gesture the user performed.

```
  private GestureDetector gestureDetector;


Override method’s are mentions here
OnDown, Scrolling, Longpress, single tap, double tab, filling
```

After all gesture implementation you don’t forget to create touch event method. Create touch event with the use of SimpleGestureFilter class object. The onTouchEvent() method of the currently active application is called by the system and passed MotionEvent objects containing data about the user’s contact with the

screen. This data can be interpreted to identify if the motion on the screen matches a common gesture such as a tap or a swipe.

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

Basic steps that detect gesture are follow as:

- Create new project. Chose blank activity write the name of activity and then press finish. Main activity class which implements the GestureDetector.OnGestureListener interface including the required onFling(), onDown(), onScroll(), onShowPress, onSingleTapUp() and onLongPress() callback methods.
- Creation of an instance of the Android GestureDetectorCompat class, passing through an instance of the class created in points one as an argument.
- If you want to required double tap so setOnDoubleTapListener() method of the GestureDetectorCompat instance to enable double tap detection.
- Implementation of the onTouchEvent() callback method on the enclosing activity which, in turn, must call the onTouchEvent() method of the GestureDetectorCompat instance, passing through the current motion event object as an argument to the method.

Once implemented, the result is a set of methods within the application code that will be called when a gesture of a particular type is detected. The code within these methods can then be implemented to perform any tasks that need to be performed in response to the corresponding gesture.

Full Implementation Example:

Step 1: first check Android manifest file.

Manifest File:

```
 <?xml version="1.0" encoding="utf-8"?><manifest
xmlns:android=http://schemas.android.com/apk/res/android


 package="com.mobilesiri.gestures">


   <application
      android:allowBackup="true"
      android:icon="@mipmap/ic_launcher"
      android:label="@string/app_name"
      android:supportsRtl="true"
      android:theme="@style/AppTheme">
      <activity android:name=".GestureActivity">
         <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
         </intent-filter>
      </activity>
   </application>



</manifest>
```

Step 2 : XML file look like that

```
 <?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.mobilesiri.gestures.GestureActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Gesture tutorial" />


 <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/imageView"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="New Text"
        android:id="@+id/textView"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true" />

</RelativeLayout>
```

Step 3: Main Activity

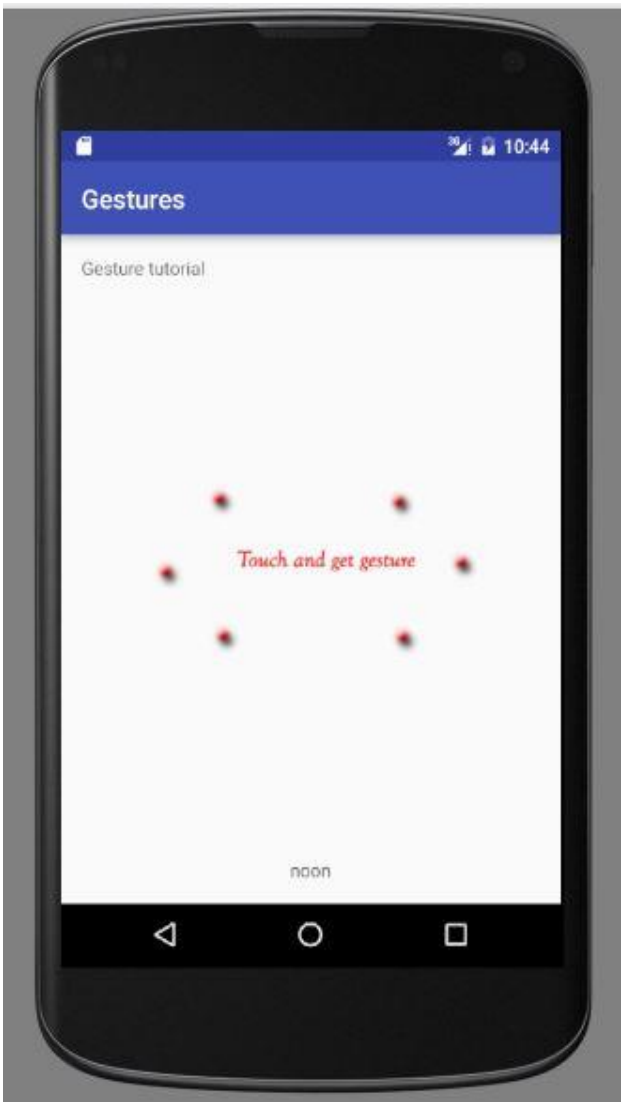
Main activity with name of Gesture implement to Gesture detect.

```
package com.mobilesiri.gestures;

import android.os.Bundle;
import android.support.v4.view.GestureDetectorCompat;
import android.support.v7.app.AppCompatActivity;
import android.view.GestureDetector;
import android.view.MotionEvent;
import android.view.View;
import android.widget.ImageView;
import android.widget.TextView;

public class GestureActivity extends AppCompatActivity implements
GestureDetector.OnGestureListener,GestureDetector.OnDoubleTapListener ,View.OnTouchListener {

    private ImageView iv;
    private TextView tv;
    private  GestureDetectorCompat gd;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_gesture);
        this.gd =new GestureDetectorCompat(this,this);
        iv = (ImageView)findViewById(R.id.imageView);
        assert iv != null;
        iv.setImageResource(R.drawable.none);
        tv=(TextView)findViewById(R.id.textView);
        tv.setText("noon");
    }


    @Override
    public boolean onDown(MotionEvent e) {
        tv.setText("On Down");
        iv.setImageResource(R.drawable.ondwn);
        return true;
    }

    @Override
    public void onShowPress(MotionEvent e) {
        tv.setText("ON show press");
        iv.setImageResource(R.drawable.onshow);

    }

    @Override
```

```
    public boolean onSingleTapUp(MotionEvent e) {
        tv.setText("single tap");
        iv.setImageResource(R.drawable.sclk);
        return true;
    }

    @Override
    public boolean onScroll(MotionEvent e1, MotionEvent e2, float distanceX, float distanceY) {
        return false;
    }

    @Override
    public void onLongPress(MotionEvent e) {

        tv.setText("long press");
        iv.setImageResource(R.drawable.longprs);

    }

    @Override
    public boolean onFling(MotionEvent e1, MotionEvent e2, float velocityX, float velocityY) {
        return false;
    }

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

    @Override
    public boolean onTouch(View v, MotionEvent event) {
        return gd.onTouchEvent(event);
    }

    @Override
    public boolean onSingleTapConfirmed(MotionEvent e) {
        return false;
    }

    @Override
    public boolean onDoubleTap(MotionEvent e) {
       //iv.setImageResource(R.drawable.dclk);
        return true;
    }

    @Override
    public boolean onDoubleTapEvent(MotionEvent e) {
        iv.setImageResource(R.drawable.dclk);
        return false;
    }
}
```

Output :

Main Design



After Detecting Gesture