## About This Article

This article provides information about how to work with Gestures in Android applications. Android application developers can use this article as reference for implementing Gestures in their applications.

### Scope:

This article is intended for Android developers wishing to develop mobile applications. It assumes basic knowledge of Android and Java programming languages. This article provides information about how to implement Gestures in Android.

To find out more about Android, please refer to the Knowledge Base under Samsung Developers.
https://developer.samsung.com/android

## Introduction

A touch screen allows a user to interact with a device in many more ways than is possible with a keypad. Users can tap, drag, fling, or slide visual elements in order to perform a variety of tasks. The Android framework makes recognizing simple tactile input easy, such as a swipe, but recognizing more complex gestures is more challenging.

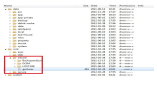
Android versions 1.6 (Donut) and later include the package *android.gesture.** for complex gesture recognition. This API stores, loads, draws, and recognizes gestures. This article explains how to use *android.gesture* in applications, including an example of how to define your own gestures and use them in an application.

## Creating Gestures

To implement gestures in an application, you first need to define the gestures. This can be done by using the *Gesture Builder* application, which is included in the Android emulator. Before creating your own gestures, make sure that the emulator has an SD card and that it is mounted. The following screen shows the *Gesture Builder* application in the emulator:

Add your own gestures and name them. The following screen shows example gestures to use in the sample application.

After creating the gestures shown above, you find the file /mnt/sdcard/gestures as shown below:



Save this file. You will use it later in the sample application.

## Create a Sample Application

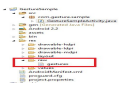
After completing the previous process, create a sample project in Eclipse and name it *GestureSample*.

## Loading a Gesture Library

To load the gesture file, use the *GestureLibraries* class. This class has functions to load data from either the SD card or a private file. The *GestureLibraries* class has the following methods:

```
static GestureLibrary fromFile(String path)
static GestureLibrary fromFile(File path)
static GestureLibrary fromPrivateFile(Context context, String name)
static GestureLibrary fromRawResource(Context context, int resourceId)
```

In this example, defined gestures are loaded from the resource folder. Copy the gestures file you saved earlier and and save it in your *GestureSample* project at **/res/raw/**. The following screen shot shows the project directory:



All of these methods return the *GestureLibrary* class, which is used to recognize, read, and save gestures. Once*GestureLibraries* returns a *GestureLibrary* class that corresponds to the file specified, all the gesture entries are read using the *GestureLibrary.load()* method.
The following code snippet shows how to load gestures file from raw folder and loading the gestures.

```
GestureLibrary gLibrary = GestureLibraries.fromRawResource(this, R.raw.gestures);

 if(gLibrary != null)
 {
   if(!gLibrary.load())
   {
     Log.e("GestureSample", "Gesture library was not loaded...");
       finish();
    }
 }
```

## Drawing and Recognizing a Gesture

To draw and recognize gestures, use the *GestureOverlayView* class. This view extends the *FrameLayout*, meaning you can use it inside any other layout or use it as a parent layout to include other child views. This view acts as an overlay view and the user can draw gestures on it. Declare the *GestureOverlayView* in the *main.xml* file. The following code snippet shows the *main.xml* file:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
         android:orientation="vertical"
         android:layout_width="fill_parent"
         android:layout_height="fill_parent"
         >
<TextView
         android:layout_width="fill_parent"
```

```
        android:layout_height="wrap_content"
        android:text="Draw your Gesture"
        />
<android.gesture.GestureOverlayView
        android:id="@+id/gestures"
        android:layout_width="fill_parent"
        android:layout_height="0dip"
        android:layout_weight="1.0" />
</LinearLayout>
```

*GestureOverlayView* uses three callback interfaces to report the actions performed, they are:

- `interface GestureOverlayView.OnGestureListener`
- `interface GestureOverlayView.OnGesturePerformedListener`
- `interface GestureOverlayView.OnGesturingListener`

These interfaces work as follows:

The *GestureOverlayView.OnGestureListener* callback interface handles the low-level gesture operations. This interface has the following methods:

```
void onGestureStarted(GestureOverlayView overlay, MotionEvent event)
void onGesture(GestureOverlayView overlay, MotionEvent event)
void onGestureEnded(GestureOverlayView overlay, MotionEvent event)
void onGestureCancelled(GestureOverlayView overlay, MotionEvent event)
```

Each of these methods has two parameters, *GestureOverlayView* and *MotionEvent*, representing the overlay view and the event that occurred, respectively.
Note the *GestureOverlayView.OnGesturePerformedListener* interface. This interface has only one method:

```
void onGesturePerformed(GestureOverlayView overlay, Gesture gesture)
```

This method is called when the user performs the gesture, which is processed by *GestureOverlayView*. The first parameter is the overlay view that is used and the second is a class Gesture that represents the user performed gesture. The Gesture class represents a hand drawn shape. This representation has one or more strokes, made up of a series of points.
The *GestureLibrary* class uses this class to recognize gestures.
The *GestureOverlayView.OnGesturingListener* callback interface is used to find when the gesture begins and ends. The interface has the following methods:

```
void onGesturingStarted(GestureOverlayView overlay)
void onGesturingEnded(GestureOverlayView overlay)
```

The *onGesturingStarted* will be called when gesture action is started and *onGesturingEnded* will be called when the gesture action ended. Both of these methods contain the *GestureOverlayView* that is used.
To recognize a gesture, use the *GestureLibrary.recognize()* method. This method accepts the *Gesture* class. This method recognizes gestures using internal recognizers and returns a list of predictions. The predictions are represented by the *Prediction* class and contain two member variables *name* and *score*. The *name* variable represents the name of the gesture and *score* variable represents the score given by the gesture recognizer. This *score* member is used to choose the best matching prediction from the list. One common method is to choose the first value that has a score greater that one. Another method is to choose a value inside a minimum and maximum threshold limit. Choosing this threshold limit depends on the implementation, ranging from simple limits based on trial and error methods, to more complex methods that may include leaning the user inputs and

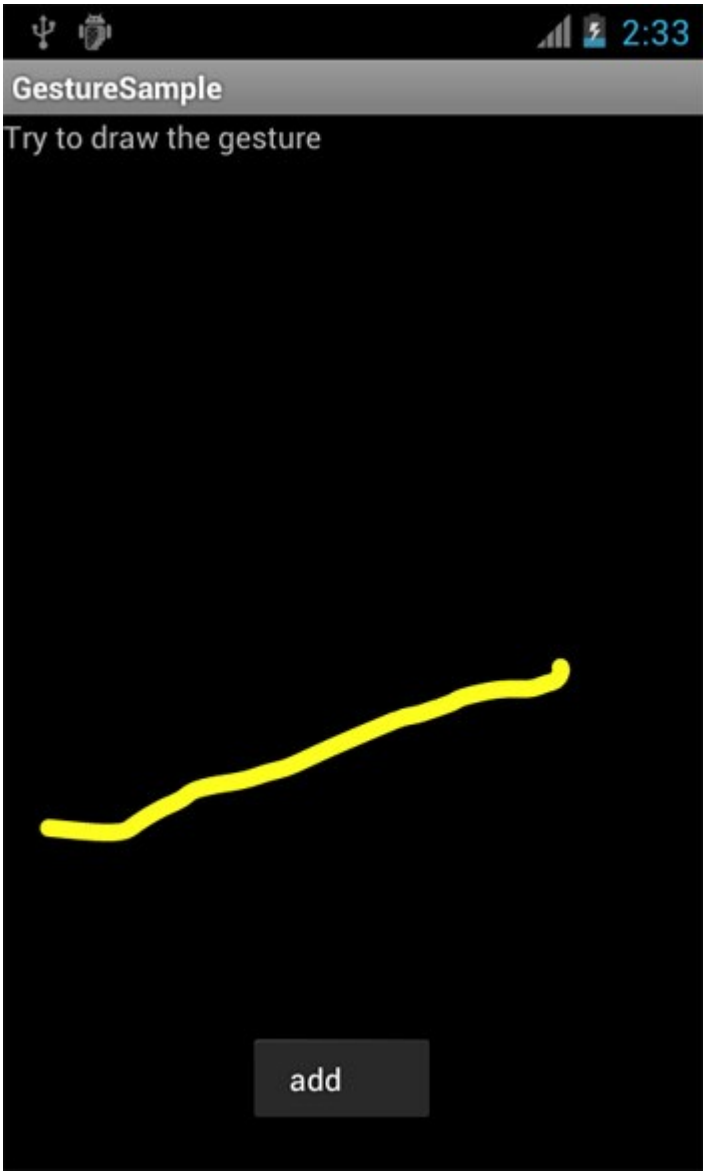
improving the recognition based on this learned user behavior.

## Code Example

The following code snippet demonstrates gesture recognition.

```
package com.gesture.sample;

import java.util.ArrayList;

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.os.Bundle;
import android.util.Log;
import android.widget.Toast;

public class GestureSampleActivity extends Activity implements OnGesturePerformedListener{

        GestureLibrary gLibrary;
        GestureOverlayView mView;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        gLibrary = GestureLibraries.fromRawResource(this, R.raw.gestures);
        if(gLibrary != null)
        {
                if(!gLibrary.load())
                {
                      Log.e("GestureSample", "Gesture library was not loaded…");
                      finish();
                }
                else
                {
                mView = (GestureOverlayView) findViewById(R.id.gestures);
                mView.addOnGesturePerformedListener(this);
                }
```

```
        }


    }

        @Override
        public void onGesturePerformed(GestureOverlayView overlay, Gesture gesture) {
                // TODO Auto-generated method stub
                ArrayList<prediction></prediction> predictions =
gLibrary.recognize(gesture);

                // one prediction needed
                if (predictions.size() > 0) {
                        Prediction prediction = predictions.get(0);
                        // checking prediction
                        if (prediction.score > 1.0) {
                                // and action
                                Toast.makeText(this,
prediction.name,Toast.LENGTH_SHORT).show();
                        }
                }
        }
}
```




Ref:http://developer.samsung.com/android/technical-docs/Gestures-in-Android