27. Detecting Common Gestures using the Android Gesture Detector Class

The term "gesture" is used to define a contiguous sequence of interactions between the touch screen and the user. A typical gesture begins at the point that the screen is first touched and ends when the last finger or pointing device leaves the display surface. When correctly harnessed, gestures can be implemented as a form of communication between user and application. Swiping motions to turn the pages of an eBook, or a pinching movement involving two touches to zoom in or out of an image are prime examples of the ways in which gestures can be used to interact with an application.

The Android SDK provides mechanisms for the detection of both common and custom gestures within an application. Common gestures involve interactions such as a tap, double tap, long press or a swiping motion in either a horizontal or a vertical direction (referred to in Android nomenclature as a *fling*).

The goal of this chapter is to explore the use of the Android GestureDetector class to detect common gestures performed on the display of an Android device. The next chapter, entitled *"Implementing Custom Gesture and Pinch Recognition on Android"*, will cover the detection of more complex, custom gestures such as circular motions and pinches.

27.1 Implementing Common Gesture Detection

When a user interacts with the display of an Android device, 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. This can be achieved with very little programming effort by making use of the Android GestureDetectorCompat class. This class is designed specifically to receive motion event information from the application and to trigger method calls based on the type of common gesture, if any, detected.

The basic steps in detecting common gestures are as follows:

1. Declaration of a class which implements the

GestureDetector.OnGestureListener interface including the required *onFling()*, *onDown()*, *onScroll()*, *onShowPress()*, *onSingleTapUp()* and *onLongPress()* callback methods. Note that this can be either an entirely new class, or the enclosing activity class. In the event that double tap gesture detection is required, the class must also implement the GestureDetector.OnDoubleTapListener interface and include the corresponding *onDoubleTap()* method.

2. Creation of an instance of the Android GestureDetectorCompat class, passing through an instance of the class created in step 1 as an argument.
3. An optional call to the *setOnDoubleTapListener()* method of the GestureDetectorCompat instance to enable double tap detection if required.
4. 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.

In the remainder of this chapter, we will work through the creation of an example project intended to put the above steps into practice.

27.2 Creating an Example Gesture Detection Project

The goal of this project is to detect the full range of common gestures currently supported by the GestureDetectorCompat class and to display status information to the user indicating the type of gesture that has been detected.

Create a new project in Android Studio, entering *CommonGestures* into the Application name field and *ebookfrenzy.com* as the Company Domain setting before clicking on the *Next* button.

On the form factors screen, enable the *Phone and Tablet* option and set the minimum SDK setting to API 14: Android 4.0 (IceCreamSandwich). Continue to proceed through the screens, requesting the creation of an Empty Activity named *CommonGesturesActivity* with a corresponding layout

resource file named *activity_common_gestures.*

Click on the *Finish* button to initiate the project creation process.

Once the new project has been created, navigate to the *app -> res -> layout -> activity_common_gestures.xml* file in the Project tool window and double-click on it to load it into the Layout Editor tool.

Within the Layout Editor tool, select the "Hello, World!" TextView component and, in the Attributes tool window, enter *gestureStatusText* as the ID.

27.3 Implementing the Listener Class

As previously outlined, it is necessary to create a class that implements the GestureDetector.OnGestureListener interface and, if double tap detection is required, the GestureDetector.OnDoubleTapListener interface. While this can be an entirely new class, it is also perfectly valid to implement this within the current activity class. For the purposes of this example, therefore, we will modify the CommonGesturesActivity class to implement these listener interfaces. Edit the *CommonGesturesActivity.java* file so that it reads as follows to declare the interfaces and to extract and store a reference to the TextView component in the user interface:

```
package com.ebookfrenzy.commongestures;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.GestureDetector;
import android.widget.TextView;

public class CommonGesturesActivity extends AppCompatActivity
        implements GestureDetector.OnGestureListener,
        GestureDetector.OnDoubleTapListener
{
    private TextView gestureText;

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

        gestureText =
                (TextView)findViewById(R.id.gestureStatusText);
```

```
    }
.
.
.
}
```

Declaring that the class implements the listener interfaces mandates that the corresponding methods also be implemented in the class:

```
package com.ebookfrenzy.commongestures;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.GestureDetector;
import android.widget.TextView;
import android.view.MotionEvent;

public class CommonGesturesActivity extends AppCompatActivity
        implements GestureDetector.OnGestureListener,
        GestureDetector.OnDoubleTapListener {

    private TextView gestureText;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_common_gestures);
        gestureText =
                (TextView)findViewById(R.id.gestureStatusText);
    }

    @Override
    public boolean onDown(MotionEvent event) {
        gestureText.setText ("onDown");
        return true;
    }

    @Override
    public boolean onFling(MotionEvent event1, MotionEvent event2,
                           float velocityX, float velocityY) {
        gestureText.setText("onFling");
        return true;
    }
```

```
    @Override
    public void onLongPress(MotionEvent event) {
        gestureText.setText("onLongPress");
    }

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

    @Override
    public void onShowPress(MotionEvent event) {
        gestureText.setText("onShowPress");
    }

    @Override
    public boolean onSingleTapUp(MotionEvent event) {
        gestureText.setText("onSingleTapUp");
        return true;
    }

    @Override
    public boolean onDoubleTap(MotionEvent event) {
        gestureText.setText("onDoubleTap");
        return true;
    }

    @Override
    public boolean onDoubleTapEvent(MotionEvent event) {
        gestureText.setText("onDoubleTapEvent");
        return true;
    }

    @Override
    public boolean onSingleTapConfirmed(MotionEvent event) {
        gestureText.setText("onSingleTapConfirmed");
        return true;
    }
.
.
```

```
.
}
```

Note that many of these methods return *true*. This indicates to the Android Framework that the event has been consumed by the method and does not need to be passed to the next event handler in the stack.

27.4 Creating the GestureDetectorCompat Instance

With the activity class now updated to implement the listener interfaces, the next step is to create an instance of the GestureDetectorCompat class. Since this only needs to be performed once at the point that the activity is created, the best place for this code is in the *onCreate()* method. Since we also want to detect double taps, the code also needs to call the *setOnDoubleTapListener()* method of the GestureDetectorCompat instance:

```
package com.ebookfrenzy.commongestures;

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

public class CommonGesturesActivity extends AppCompatActivity
        implements GestureDetector.OnGestureListener,
        GestureDetector.OnDoubleTapListener {

    private TextView gestureText;
    private GestureDetectorCompat gDetector;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_common_gestures);
        gestureText =
                (TextView)findViewById(R.id.gestureStatusText);

        this.gDetector = new GestureDetectorCompat(this,this);
        gDetector.setOnDoubleTapListener(this);
    }
.
.
```

```
}
```

27.5 Implementing the onTouchEvent() Method

If the application were to be compiled and run at this point, nothing would happen if gestures were performed on the device display. This is because no code has been added to intercept touch events and to pass them through to the GestureDetectorCompat instance. In order to achieve this, it is necessary to override the *onTouchEvent()* method within the activity class and implement it such that it calls the *onTouchEvent()* method of the GestureDetectorCompat instance. Remaining in the *CommonGesturesActivity.java* file, therefore, implement this method so that it reads as follows:

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

27.6 Testing the Application

Compile and run the application on either a physical Android device or an AVD emulator. Once launched, experiment with swipes, presses, scrolling motions and double and single taps. Note that the text view updates to reflect the events as illustrated in Figure 27-1:

Figure 27-1

27.7 Summary

Any physical contact between the user and the touch screen display of a device can be considered a “gesture”. Lacking the physical keyboard and mouse pointer of a traditional computer system, gestures are widely used as a method of interaction between user and application. While a gesture can be comprised of just about any sequence of motions, there is a widely used set of gestures with which users of touch screen devices have become familiar. A number of these so-called “common gestures” can be easily detected within an application by making use of the Android Gesture Detector classes. In this chapter, the use of this technique has been outlined both in theory and through the implementation of an example project.

Having covered common gestures in this chapter, the next chapter will look at detecting a wider range of gesture types including the ability to both design and detect your own gestures.