MODULE

Multi-Touch & Gestures Detectors

Nawfal Ali

Updated 27 May 2020

Q) When does the multi-touch gesture happen?

A) It happens when more than one pointer (finger) touches the screen.

Q) How to keep track of each pointer within a gesture?

A) You have to use the pointer's index and ID.

Q) What is the purpose of having an Index and ID for each pointer?

A) Each pointer gets a unique ID during the gesture's lifetime and it is used to track the pointers within the gesture. This ID is generated once the pointer touches the screen and joins the gesture.

Now, the MotionEvent object saves all the pointers' data in a special array and uses indices to access the pointers' entries. These entries might shift up (change) if a pointer leaves the screen (the gesture) and this will lead to changes in pointers' indices.

Reference: https://developer.android.com/training/gestures/multi

Android system generates touch events every time multiple pointers touches the screen at the same time: (Click HERE for more details)

EVENT	DESCRIPTION
ACTION_DOWN	The first pointer (finger) touches the screen. The motion event object contains the initial st
ACTION_POINTER_UP	A non-primary (secondary) pointer leaves the screen
ACTION_POINTER_DOWN	A non-primary (secondary) pointer touches the screen
ACTION_MOVE	A motion happened to primary or non-primary pointer
ACTION_UP	The last pointer leaves the screen

Android system provides several methods (MotionEvent methods) to deal with multi-touch events, such as:

METHOD	DESCRIPTION
getPointerCount()	The current number of pointers (fingers) on the screen

METHOD	DESCRIPTION
getPointerId(int pointerIndex)	get the pointer id associated with a particular pointer data index in the current gesture
findPointerIndex(int pointerId)	find the pointer index for the given id
getX(int pointerIndex)	find the x coordinate for the given pointer index
getY(int pointerIndex)	find the y coordinate for the given pointer index

Gesture Detectors

The gesture detector classes are used to detect common gestures through a set of motion events.

There are three steps required for the gesture detector to work:

1. Create an instance of Gesture Detector class
2. Implements the required methods
3. Intercept the touch events and pass them to the gesture detector

Mainly, there are two classes of detectors GestureDetector and ScaleGestureDetector

CLASS	INTERFACE	METHODS	DESCRIPTION
GestureDetector	OnGestureListener	onDown(MotionEvent e)	Notified when a tap occurs with the down MotionEvent th
		onFling(MotionEvent e1, MotionEvent e2, float velocityX, float velocityY)	- Notified of a fling event when it occurs with - e1 is the first event (touch down) - e2 is the motion event that triggered the cur - velocityX: The velocity of this event (fling ev - velocityY: The velocity of this event (fling ev
		onLongPress(MotionEvent e)	Notified when a long press occurs with the initial on dowr
		onScroll(MotionEvent e1, MotionEvent e2, float distanceX, float distanceY)	- Notified when a scroll occurs with the initial - e1 is the first event (touch down) - e2 is the current event - **distanceX: the distance between e2 and the** - **distanceX: the distance between e2 and the**
		onShowPress(MotionEvent e)	The user has performed a down MotionEvent and not perf
		onSingleTapUp(MotionEvent e)	Notified when a tap occurs with the up MotionEvent that

CLASS	INTERFACE	METHODS	DESCRIPTION
GestureDetector	OnDoubleTapListener	onDoubleTap(MotionEvent e)	Notified when a double-tap occurs.
		onDoubleTapEvent(MotionEvent e)	Notified when an event within a double-tap gesture occur
		onSingleTapConfirmed(MotionEvent e)	Notified when a single-tap occurs.

CLASS	INTERFACE	METHOD
ScaleGestureDetector	OnScaleGestureListener	onScale(ScaleGestureDetector detector)
		onScaleBegin(ScaleGestureDetector detector)
		onScaleEnd(ScaleGestureDetector detector)

Q) Some methods return boolean values, what does that mean?

A) If a callback method returns true, it informs the parent that the event has been consumed and ready to accept further events from the current gesture. But, if a callback returns false, it indicates that the event is not consumed and it is not interested in the remainder of the gesture.

Q) What are the differences between onFling() and onScroll() callbacks?

A) 1) OnFling() needs some velocity in the movement (like swipe to unlock the phone). While, onScroll(), is invoked one you move your finger with normal speed (when you scroll a list or drag and drop)

2) onFlicg() will be called only once at the end of the gesture, while onScroll() will be called multiple times as you move your finger on the screen.

Instantiate a Gesture Detector instance

```java
private GestureDetectorCompat mDetector;
private ScaleGestureDetector mScaleDetector;

...
...
mDetector = new GestureDetectorCompat(this, this);
mScaleDetector = new ScaleGestureDetector(this, this);
```

Where the first parameter is the context and the second one is a reference to the callbacks object.

Implements the required methods

It is possible to create a separate class to handle the implementation of all the callbacks or simply add them to the activity as shown below:

```java
public class MainActivity extends AppCompatActivity implements GestureDetector.OnGestureListener, GestureDetector.OnDoubleTapListener {

}
```

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

   @Override
   public void onShowPress(MotionEvent e) {

   }

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

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

   @Override
   public void onLongPress(MotionEvent e) {
      return false;

   }

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

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

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

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


   @Override
   public boolean onScale(ScaleGestureDetector detector) {
       return false;
```

```
    }

    @Override
    public boolean onScaleBegin(ScaleGestureDetector detector) {
        return true;
    }
    @Override
    public void onScaleEnd(ScaleGestureDetector detector) {

    }
```

Intercept the touch events and pass them to the gesture detector

For the gesture detectors to work, you must override the onTouch callback method and forward the MotionEvent object to the detectors.

> Q) What will happen if your onTouch() callback returns false instead of true?
>
> Change it and observe the results.

```
@Override
public boolean onTouch(View v, MotionEvent event) {
       mDetector.onTouchEvent(event));
       mScaleDetector.onTouchEvent(event);

       return true;
    }
```

Now, the activity in one piece:

```
package com.fit2081.week11gesturesdetectors;

import androidx.appcompat.app.AppCompatActivity;
import androidx.core.view.GestureDetectorCompat;

import android.os.Bundle;
import android.view.GestureDetector;
import android.view.MotionEvent;
import android.view.ScaleGestureDetector;
import android.view.View;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity implements View.OnTouchListener,
GestureDetector.OnGestureListener, GestureDetector.OnDoubleTapListener, ScaleGestureDetector.OnScaleGestureListener {

    TextView tV;
    private GestureDetectorCompat mDetector;
    private ScaleGestureDetector mScaleDetector;

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

        tV = findViewById(R.id.textview_id);
        mScaleDetector = new ScaleGestureDetector(this, this);

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

        View myLayout = findViewById(R.id.myLayout);
        myLayout.setOnTouchListener(this);


    }

    @Override
    public boolean onTouch(View v, MotionEvent event) {
        mDetector.onTouchEvent(event);
        mScaleDetector.onTouchEvent(event);

        return true;
    }

    @Override
    public boolean onSingleTapConfirmed(MotionEvent e) {
        tV.setText("onSingleTapConfirmed");
        return true;
    }

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

    @Override
    public boolean onDoubleTapEvent(MotionEvent e) {
        tV.setText("onDoubleTapEvent");

        return true;
    }

    @Override
    public boolean onDown(MotionEvent e) {
        tV.setText("onDown");

        return true;
    }

    @Override
    public void onShowPress(MotionEvent e) {
        tV.setText("onShowPress");

    }

    @Override
    public boolean onSingleTapUp(MotionEvent e) {
        tV.setText("onSingleTapUp");

        return true;
    }

    @Override
    public boolean onScroll(MotionEvent e1, MotionEvent e2, float distanceX, float distanceY) {
        tV.setText("onScroll");

        return true;
    }

    @Override
    public void onLongPress(MotionEvent e) {
        tV.setText("onLongPress");

    }

    @Override
    public boolean onFling(MotionEvent e1, MotionEvent e2, float velocityX, float velocityY) {
        tV.setText("onFling");

        return true;
    }


    @Override
    public boolean onScale(ScaleGestureDetector detector) {
        tV.setText("onScale");

        return true;
    }

    @Override
    public boolean onScaleBegin(ScaleGestureDetector detector) {
        tV.setText("onScaleBegin");

        return true;
    }

    @Override
    public void onScaleEnd(ScaleGestureDetector detector) {
        tV.setText("onScaleEnd");

    }
}
```

Q) What will happen if onScaleBegin() callback returns false instead of true?

A) The two methods onScale() and onScaleEnd() will not be invoked.

Q) Why?

A) if the callback onScaleBegin() returns false, this means it is not interested in the current gesture.

Q) How can I get the pinch (zoom/scale) size at the end of the scale gesture?

A) you can return the event span using **detector.getCurrentSpan()** where detector is the input parameter to onScaleEnd() callback.

Click HERE for more information about the ScaleGestureDetector.

Q) What if want to implement only a subset of the callback? in other words, my app requires one or two callbacks only.

A) You need to implement a class that extends the convenience classes SimpleOnGestureListener or SimpleOnScaleGestureListener.

Build a class the extends SimpleOnGestureListener

```
private class MyGestureListener extends GestureDetector.SimpleOnGestureListener {

        @Override
        public boolean onSingleTapConfirmed(MotionEvent e) {
            tV.setText("onSingleTapConfirmed");
            return true;
        }

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

        @Override
        public boolean onDown(MotionEvent e) {
            tV.setText("onDown");

            return true;
        }

        @Override
        public boolean onScroll(MotionEvent e1, MotionEvent e2, float distanceX, float distanceY) {
            tV.setText("onScroll");

            return true;
        }

        @Override
        public void onLongPress(MotionEvent e) {
            tV.setText("onLongPress");

        }
    }
```

As you can see from the code above, the class MyGestureListener which works as a listener implements a subset of the interface OnGestureListener callbacks.

Build a class the extends SimpleOnScaleGestureListener

```
private class MyScaleListener extends ScaleGestureDetector.SimpleOnScaleGestureListener{
    @Override
    public boolean onScale(ScaleGestureDetector detector) {
        tV.setText("onScale");

        return true;
    }

    @Override
    public boolean onScaleBegin(ScaleGestureDetector detector) {
        tV.setText("onScaleBegin");

        return true;
    }

}
```

Again, extending the convenience class SimpleOnScaleGestureListener allows you to implement only the callbacks which your application needs.

Now, let's provide them to the detectors:

```
MyScaleListener MyScaleListener=new MyScaleListener();
mScaleDetector = new ScaleGestureDetector(this, MyScaleListener);

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

Disclaimer and Copyright

Privacy

Service Status