ANDROID PROGRAMMING LESSON 5

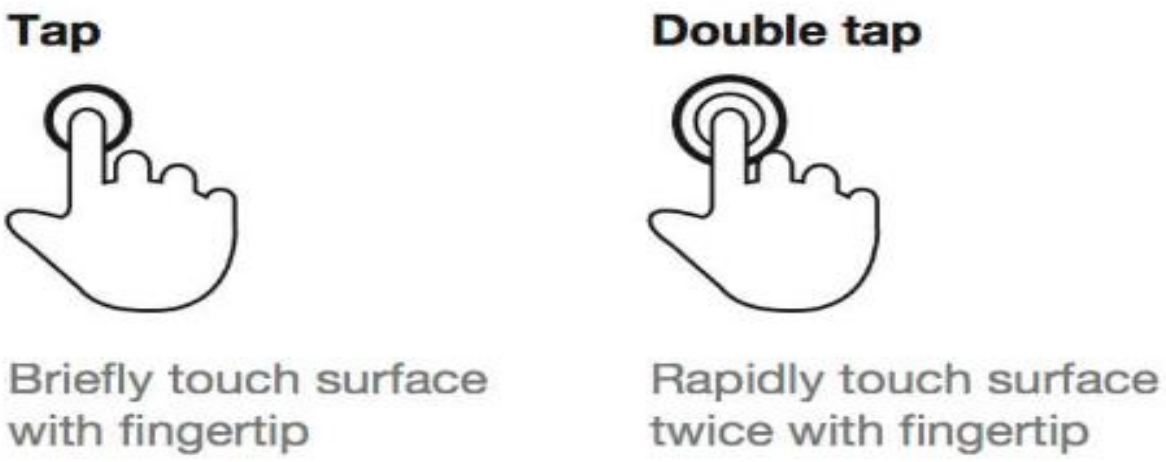
Version 1.0

Agenda

- Android Touch and Multi-touch Event Handling
- An Example Multi-touch app (01)
- Detecting Common Gestures using the Android GestureDetector Class
- Drag/Move views around the screen (02)
- Implementing Custom Gesture and Pinch Recognition on Android
- ScaleGestureDetector Demo(03)

Android Touch and Multi-touch Event Handling

- Most Android based devices use a touch screen as the primary interface between user and device.
- Much more to touch event handling than responding to a single finger tap on a view object



Android Touch and Multi-touch Event Handling

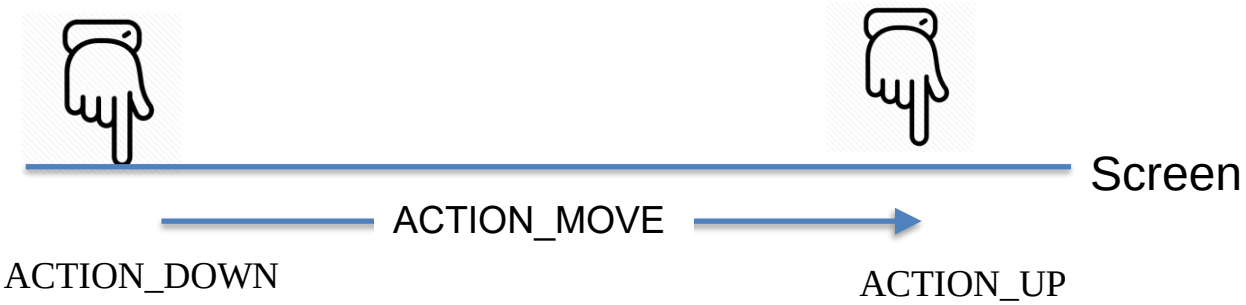
- **Intercepting Touch Events**

  Touch events can be intercepted by a view object through the registration of an *onTouchListener* event listener and the implementation of the corresponding *onTouch()* callback method

```
myLayout.setOnTouchListener(
        new ConstraintLayout.OnTouchListener() {
            public boolean onTouch(View v, MotionEvent m) {
                // -------------- TO DO -----------------
                return true;
            }
        }
);
```

Android Touch and Multi-touch Event Handling

- **The MotionEvent Object**
  The MotionEvent object passed through to the *onTouch()* callback method is the key to obtaining information about the event
- **Understanding Touch Actions**



When more than one touch is performed simultaneously on a view, the touches are referred to as *pointers*. Pointers begin and end with event actions of type ACTION_POINTER_DOWN and ACTION_POINTER_UP

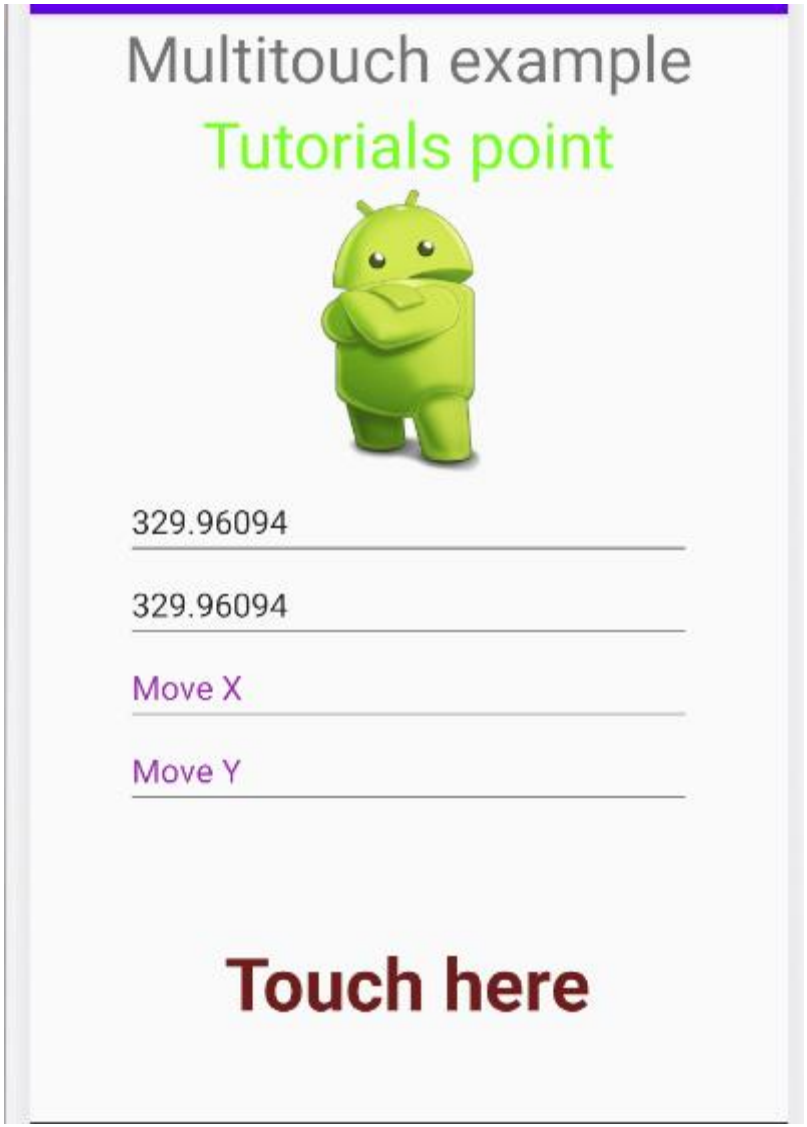
Sr.No	Event & description
1	**ACTION_DOWN** For the first pointer that touches the screen. This starts the gesture.
2	**ACTION_POINTER_DOWN** For extra pointers that enter the screen beyond the first.
3	**ACTION_MOVE** A change has happened during a press gesture.
4	**ACTION_POINTER_UP** Sent when a non-primary pointer goes up.
5	**ACTION_UP** Sent when the last pointer leaves the screen.

Android Touch and Multi-touch Event Handling

- So in order to detect any of the above mention event , you need to override **onTouchEvent()** method and check these events manually. Its syntax is given below

```
public boolean onTouchEvent(MotionEvent ev){
    final int actionPeformed = ev.getAction();
    switch(actionPeformed){
       case MotionEvent.ACTION_DOWN:{
                                    break; }
       case MotionEvent.ACTION_MOVE:{
                                    break; }
     return true;
 }
}
```

An Example Multi-touch app (01)



```
public class MainActivity extends AppCompatActivity {
    private EditText ed1, ed2, ed3, ed4;
    private TextView tv1;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        initView();
        tv1.setOnTouchListener(new View.OnTouchListener() {
            @Override
            public boolean onTouch(View v, MotionEvent event)
{
                int actionPeformed = event.getAction();
                switch(actionPeformed){
                    case MotionEvent.ACTION_DOWN:{
                        float x = event.getX();
                        float y = event.getY();
                        ed1.setText(Float.toString(x));
                        ed2.setText(Float.toString(x));
                        break;
                    }
```

```
case MotionEvent.ACTION_MOVE:{
                        float x = event.getX();
                        float y = event.getY();
                        ed3.setText(Float.toString(x));
                        ed4.setText(Float.toString(y));
                        break;
                    }
                }
                return true;
            }
        });
    }
    private void initView() {
        ed1 = (EditText) findViewById(R.id.editText);
        ed2 = (EditText) findViewById(R.id.editText2);
        ed3 = (EditText) findViewById(R.id.editText3);
        ed4 = (EditText) findViewById(R.id.editText4);
        tv1=(TextView)findViewById(R.id.textView2);
    }
}
```

GestureDetector class

- Learn how to detect basic touch gestures such as scrolling, flinging, and double-tapping, using GestureDetector.
- include 3 interface Listener:
  - GestureDetector.OnGestureListener
  - GestureDetector.OnDoubleTapListener
  - GestureDetector.OnContextClickListener

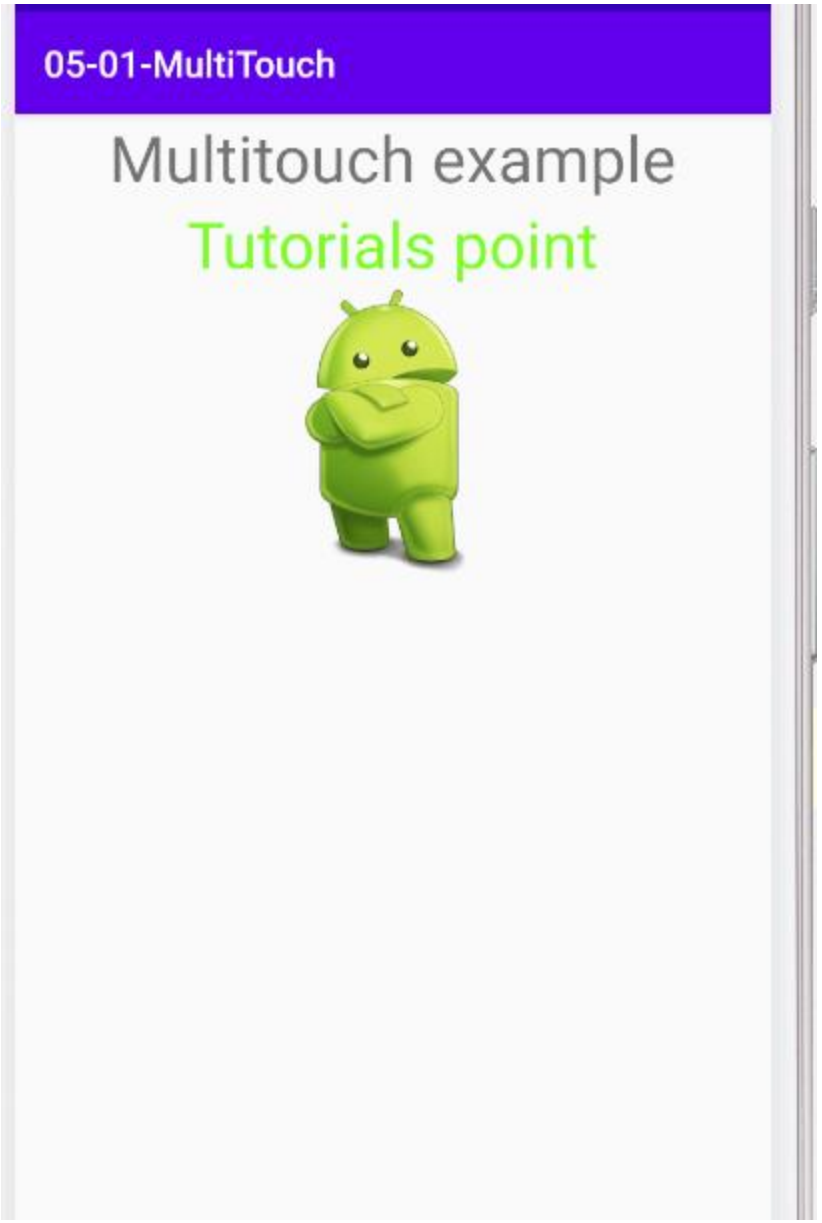
OnGestureListener interface

- boolean onDown(MotionEvent e);
- void onShowPress(MotionEvent e);
- boolean onSingleTapUp(MotionEvent e);
- boolean onScroll(MotionEvent e1, MotionEvent e2, float distanceX, float distanceY);
- void onLongPress(MotionEvent e);
- boolean onFling(MotionEvent e1, MotionEvent e2, float velocityX, float velocityY);

OnDoubleTapListener, OnContextClickListener interface

- boolean onSingleTapConfirmed(MotionEvent e);
- boolean onDoubleTap(MotionEvent e);
- boolean onDoubleTapEvent(MotionEvent e);
- boolean onContextClick(MotionEvent e);

Drag/Move views around the screen (02)



Thực thi lớp lắng nghe sự kiện (listener class)

- Tạo lớp thực thi giao diện GestureDetector.OnGestureListener và GestureDetector.OnDoubleTapListener (cho sự kiện double tap).

```
import android.view.GestureDetector;
public class MainActivity extends
AppCompatActivity
        implements
GestureDetector.OnGestureListener,
        GestureDetector.OnDoubleTapListener,
        View.OnTouchListener{
```

```
public class MainActivity extends AppCompatActivity
        implements GestureDetector.OnGestureListener,
        GestureDetector.OnDoubleTapListener,
        View.OnTouchListener{
    private TextView tv1,tv2,tv3,tv4,tv5,tv6;
    private ImageView image1;
    private GestureDetector gestureDetector;
    @Override
    protected void onCreate(Bundle savedInstanceState)
{
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        initView();
        image1.setOnTouchListener(this);
        gestureDetector=new GestureDetector(this,
                this);
        gestureDetector.setOnDoubleTapListener(this);
    }
```

```
// methods of OnGestureListener interface
@Override
public boolean onDown(MotionEvent e) {
    tv3.setText("down touch");
    tv4.setText("x:"+e.getX()+" and y:"+e.getY());
    return true;
}
@Override
public void onShowPress(MotionEvent e) {
    tv3.setText("Show press");
    tv4.setText("x:"+e.getX()+" and y:"+e.getY());
}
@Override
public boolean onSingleTapUp(MotionEvent e) {
    tv5.setText("Single tap up");
    tv6.setText("x:"+e.getX()+" and y:"+e.getY());
    return true;
}
@Override
public boolean onScroll(MotionEvent e1, MotionEvent e2, float
distanceX, float distanceY) {
    tv5.setText("Scroll");
    tv6.setText("x:"+e1.getX()+" and y:"+e1.getY());
    return true;
}
```

```
@Override
public void onLongPress(MotionEvent e) {
    tv5.setText("Long press");
    tv6.setText("x:"+e.getX()+" and y:"+e.getY());
}
@Override
public boolean onFling(MotionEvent e1, MotionEvent e2,
float velocityX, float velocityY) {
    tv1.setText("Fling");
    tv2.setText("x1:"+e1.getX()+" and y1:"+e1.getY());
    return true;
}
```

```
// methods of OnDoubleTapListener interface
@Override
public boolean onSingleTapConfirmed(MotionEvent e) {
    tv1.setText("single tap after double tap");
    tv2.setText("x:"+e.getX()+" and y:"+e.getY());
    return true;
}
@Override
public boolean onDoubleTap(MotionEvent e) {
    tv1.setText("double tap");
    tv2.setText("x:"+e.getX()+" and y:"+e.getY());
    return true;
}
@Override
public boolean onDoubleTapEvent(MotionEvent e) {
    tv1.setText("onDoubleTapEventdouble tap event");
    tv2.setText("x:"+e.getX()+" and y:"+e.getY());
    return true;
}
```

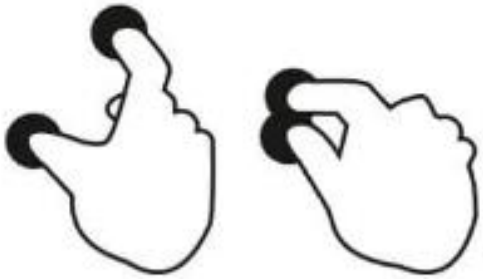
- Các phương thức trả về giá trị true để xác nhận các sự kiện có thể được xử lý bởi các phương thức mà không cần chuyển đến hàng đợi xử lý sự kiện.
- **Thực thi phương thức onTouchEvent()**
- Bước cuối cùng là thực thi phương thức onTouchEvent() trong lớp MainActivity như sau:

```
//OnTouchListener interface
@Override
public boolean onTouch(View v, MotionEvent event) {
    gestureDetector.onTouchEvent(event);
    return true;
}
```

Implementing Custom Gesture and Pinch Recognition

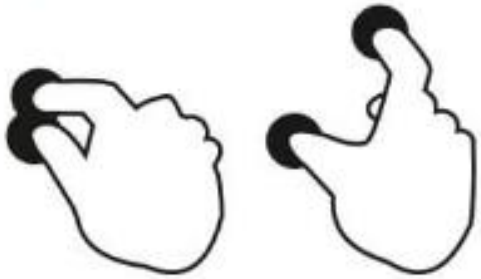
- The concept behind the pinch gesture is simple.
  - Are there 2 or more fingers on the screen?
  - Is the gap between the fingers shrinking or expanding?
- Detecting a pinch
  - Detecting a pinch gesture is similar to a scroll gesture, we simply want to make sure that there are 2 fingers, they have moved beyond a certain threshold and that they are moving in opposite directions.

Pinch



Touch surface with two fingers and bring them closer together

Spread

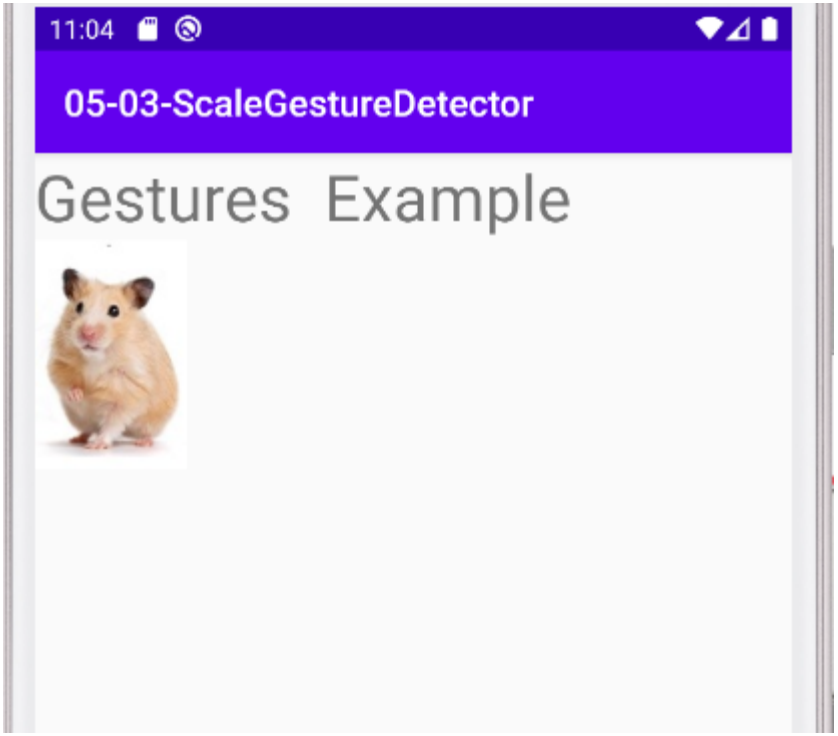


Touch surface with two fingers and move them apart

Press

Touch surface for extended period of time

ScaleGestureDetector demo (03)



Xử lý Pinch Gesture

```
public class MainActivity extends AppCompatActivity {
        private ImageView iv;
        private Matrix matrix = new Matrix();
        private float scale = 1f;
        private ScaleGestureDetector SGD;
    @Override
    protected void onCreate(Bundle
savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        iv=(ImageView)findViewById(R.id.image1);
        if(SGD==null)
           SGD = new ScaleGestureDetector(this,new
ScaleListener());
    }
```

```
public boolean onTouchEvent(MotionEvent ev) {
            SGD.onTouchEvent(ev);
            return true;
        }
private class ScaleListener extends
ScaleGestureDetector.SimpleOnScaleGestureListener {
            @Override
    public boolean onScale(ScaleGestureDetector
        detector) {
        scale *= detector.getScaleFactor();
        scale = Math.max(0.1f, Math.min(scale,
10.0f));
        matrix.setScale(scale, scale);
        iv.setImageMatrix(matrix);
        return true;
              }
        }
}
```

- End of Lesson 5

Thank you!