

タッチイベントを取得する

タッチパネルを操作すると、タッチイベントが ACTION_DOWN→ACTION_MOVE(繰返し)→ACTION_UP の順に発生する。

このタッチイベントを取得するには、onTouchEvent メソッドをオーバーライドする。

また、dispatchTouchEvent メソッドをオーバーライドしても、同様の情報を取得することができる。dispatchTouchEvent メソッドは、onTouchEvent メソッドよりも先にコールされる。

アクティビティで取得する

- Activity#onTouchEvent メソッドをオーバーライドする。
- MotionEvent#getX メソッド, getY メソッドで、タッチされた x, y 座標を取得できる。
- MotionEvent#.getAction メソッドで、タッチアクション (DOWN/MOVE/UP/CANCEL)を取得できる。
- MotionEvent#getTime メソッドで、イベント発生時刻(ms)を取得できる。(サンプルコードでは未使用)

```
package sample.android.touchevent01;
```

```
import android.app.Activity;
```

```
import android.os.Bundle;
```

```
import android.util.Log;
```

```
import android.view.MotionEvent;
```

```
public class TouchEvent01 extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

```
@Override
```

```
public boolean onTouchEvent(MotionEvent event) {
```

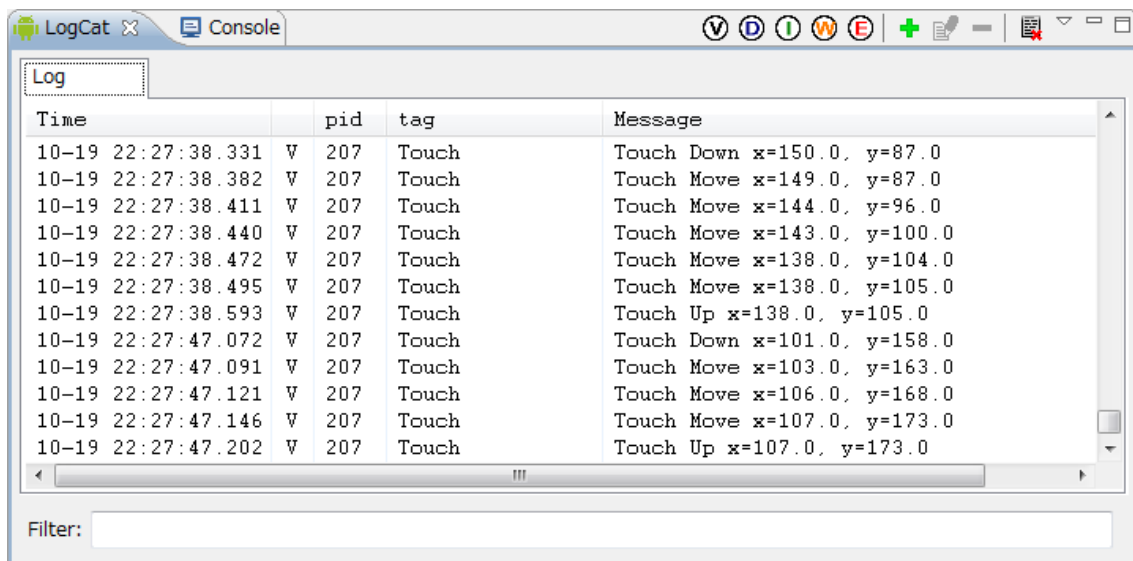
```

// TODO Auto-generated method stub
float x = event.getX();
float y = event.getY();
String action = "";

switch(event.getAction()) {
case MotionEvent.ACTION_DOWN:
    action = "Touch Down";
    break;
case MotionEvent.ACTION_MOVE:
    action = "Touch Move";
    break;
case MotionEvent.ACTION_UP:
    action = "Touch Up";
    break;
case MotionEvent.ACTION_CANCEL:
    action = "Touch Cancel";
    break;
}

Log.v("Touch", action + " x=" + x + ", y=" + y);
return super.onTouchEvent(event);
}
}

```



View で取得する

- View#onTouchEvent メソッドをオーバーライドする.
 - MotionEvent#getX メソッド, getY メソッドで, タッチされた x, y 座標を取得できる.
 - MotionEvent#getTime メソッドや getEventTime メソッドも Activity 同様使用できる.
-

```
package sample.android.touchevent02;

import java.util.ArrayList;

import android.app.Activity;
import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.Point;
import android.os.Bundle;
import android.view.MotionEvent;
import android.view.View;

public class TouchEvent02 extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        MyView view = new MyView(this);
        setContentView(view);
    }
}

class MyView extends View {
    private ArrayList<Point> points = new ArrayList<Point>();
```

```

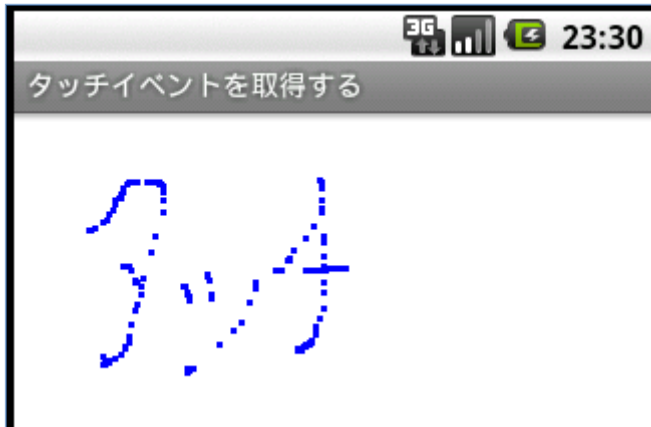
public MyView(Context context) {
    super(context);
    setFocusable(true);
}

@Override
protected void onDraw(Canvas canvas) {
    // TODO Auto-generated method stub
    canvas.drawColor(Color.WHITE);
    Paint paint = new Paint();
    paint.setColor(Color.BLUE);
    paint.setStrokeWidth(3);
    Point po = null;
    for(int i=0; i<points.size(); i++) {
        po = points.get(i);
        canvas.drawPoint(po.x, po.y, paint);
    }
    super.onDraw(canvas);
}

@Override
public boolean onTouchEvent(MotionEvent event) {
    // TODO Auto-generated method stub
    int x = (int)event.getX();
    int y = (int)event.getY();
    points.add(new Point(x, y));
    invalidate(); // 通知
    return true;
}
}

```

● 実行結果

クリックイベント、タッチイベント、キーイベントの同居

```
package sample.android.multievent;
```

```
import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
import android.view.KeyEvent;
import android.view.MotionEvent;
import android.view.Window;
import android.widget.Button;
import android.widget.LinearLayout;
import android.widget.Toast;
import android.view.View;
```

```
public class MultiEventSample extends Activity implements View.OnClickListener {
```

```
    /** Called when the activity is first created. */
```

```
    @Override
```

```
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
        LinearLayout linearLayout = new LinearLayout(this);
```

```

linearLayout.setOrientation(LinearLayout.VERTICAL);
setContentView(linearLayout);

Button button1 = new Button(this);
button1.setText("Yes");
button1.setOnClickListener(this); // ボタンにクリックリスナー設定
linearLayout.addView(button1, new LinearLayout.LayoutParams(
    LinearLayout.LayoutParams.WRAP_CONTENT,
    LinearLayout.LayoutParams.WRAP_CONTENT));

}

// 画面タッチイベント発生時、呼び出されます
@Override
public boolean onTouchEvent(MotionEvent event) {
    String action = "";
    switch (event.getAction()) {
        case MotionEvent.ACTION_DOWN:
            action = "ACTION_DOWN";
            break;
        case MotionEvent.ACTION_UP:
            action = "ACTION_UP";
            break;
        case MotionEvent.ACTION_MOVE:
            action = "ACTION_MOVE";
            break;
        case MotionEvent.ACTION_CANCEL:
            action = "ACTION_CANCEL";
            break;
    }
    // ログ情報を LogCat 画面に出力
    Log.i("MotionEvent", "action = " + action + ", " + "x = "
        + String.valueOf(event.getX()) + ", " + "y = "
        + String.valueOf(event.getY()));
    return super.onTouchEvent(event);
}

```

```

// キーイベント発生時、呼び出されます
public boolean dispatchKeyEvent(KeyEvent event) {
    if (event.getAction() == KeyEvent.ACTION_UP) { // キーが離された時
        switch (event.getKeyCode()) {
            case KeyEvent.KEYCODE_DPAD_CENTER: // 十字中央キー

                Toast.makeText(this, "十字中央キー", Toast.LENGTH_SHORT).show();
                return true;

            case KeyEvent.KEYCODE_DPAD_RIGHT: // 十字右キー
                Toast.makeText(this, "十字右キー", Toast.LENGTH_SHORT).show();
                return true;
            default:
            }
        }
        return super.dispatchKeyEvent(event);
    }

    // ボタンをタッチしたときの処理
    public void onClick(View arg0) {
        Toast.makeText(this, "ボタンをタッチ", Toast.LENGTH_SHORT).show();
    }
}

```
