# [Android手势识别（单击 双击 抬起 短按 长按 滚动 滑动）](http://www.cnblogs.com/Free-Thinker/p/4795112.html)

对于触摸屏，其原生的消息无非按下、抬起、移动这几种，我们只需要简单重载onTouch或者设置触摸侦听器setOnTouchListener即可进行处理。不过，为了提高我们的APP的用户体验，有时候我们需要识别用户的手势，Android给我们提供的手势识别工具GestureDetector就可以帮上大忙了。

* 基础

GestureDetector的工作原理是，当我们接收到用户触摸消息时，将这个消息交给GestureDetector去加工，我们通过设置侦听器获得GestureDetector处理后的手势。

GestureDetector提供了两个侦听器接口，OnGestureListener处理单击类消息，OnDoubleTapListener处理双击类消息。

OnGestureListener的接口有这几个：

1. // 单击，触摸屏按下时立刻触发
2. abstract boolean onDown(MotionEvent e);
3. // 抬起，手指离开触摸屏时触发(长按、滚动、滑动时，不会触发这个手势)
4. abstract boolean onSingleTapUp(MotionEvent e);
5. // 短按，触摸屏按下后片刻后抬起，会触发这个手势，如果迅速抬起则不会
6. abstract void onShowPress(MotionEvent e);
7. // 长按，触摸屏按下后既不抬起也不移动，过一段时间后触发
8. abstract void onLongPress(MotionEvent e);
9. // 滚动，触摸屏按下后移动
10. abstract boolean onScroll(MotionEvent e1, MotionEvent e2, float distanceX, float distanceY);
11. // 滑动，触摸屏按下后快速移动并抬起，会先触发滚动手势，跟着触发一个滑动手势
12. abstract boolean onFling(MotionEvent e1, MotionEvent e2, float velocityX, float velocityY);

OnDoubleTapListener的接口有这几个：

1. // 双击，手指在触摸屏上迅速点击第二下时触发
2. abstract boolean onDoubleTap(MotionEvent e);
3. // 双击的按下跟抬起各触发一次
4. abstract boolean onDoubleTapEvent(MotionEvent e);
5. // 单击确认，即很快的按下并抬起，但并不连续点击第二下
6. abstract boolean onSingleTapConfirmed(MotionEvent e);

有时候我们并不需要处理上面所有手势，方便起见，Android提供了另外一个类SimpleOnGestureListener实现了如上接口，我们只需要继承SimpleOnGestureListener然后重载感兴趣的手势即可。

* 应用

STEP 1: 创建手势侦听对象

1. package noodies.blog.csdn.net;
3. import android.content.Context;
4. import android.view.MotionEvent;
5. import android.view.GestureDetector.SimpleOnGestureListener;
6. import android.widget.Toast;
8. public class MyGestureListener extends SimpleOnGestureListener {
10. private Context mContext;
12. MyGestureListener(Context context) {
13. mContext = context;
14. }
16. @Override
17. public boolean onDown(MotionEvent e) {
18. Toast.makeText(mContext, "DOWN " + e.getAction(), Toast.LENGTH\_SHORT).show();
19. return false;
20. }
22. @Override
23. public void onShowPress(MotionEvent e) {
24. Toast.makeText(mContext, "SHOW " + e.getAction(), Toast.LENGTH\_SHORT).show();
25. }
27. @Override
28. public boolean onSingleTapUp(MotionEvent e) {
29. Toast.makeText(mContext, "SINGLE UP " + e.getAction(), Toast.LENGTH\_SHORT).show();
30. return false;
31. }
33. @Override
34. public boolean onScroll(MotionEvent e1, MotionEvent e2,
35. float distanceX, float distanceY) {
36. Toast.makeText(mContext, "SCROLL " + e2.getAction(), Toast.LENGTH\_SHORT).show();
37. return false;
38. }
40. @Override
41. public void onLongPress(MotionEvent e) {
42. Toast.makeText(mContext, "LONG " + e.getAction(), Toast.LENGTH\_SHORT).show();
43. }
45. @Override
46. public boolean onFling(MotionEvent e1, MotionEvent e2, float velocityX,
47. float velocityY) {
48. Toast.makeText(mContext, "FLING " + e2.getAction(), Toast.LENGTH\_SHORT).show();
49. return false;
50. }
52. @Override
53. public boolean onDoubleTap(MotionEvent e) {
54. Toast.makeText(mContext, "DOUBLE " + e.getAction(), Toast.LENGTH\_SHORT).show();
55. return false;
56. }
58. @Override
59. public boolean onDoubleTapEvent(MotionEvent e) {
60. Toast.makeText(mContext, "DOUBLE EVENT " + e.getAction(), Toast.LENGTH\_SHORT).show();
61. return false;
62. }
64. @Override
65. public boolean onSingleTapConfirmed(MotionEvent e) {
66. Toast.makeText(mContext, "SINGLE CONF " + e.getAction(), Toast.LENGTH\_SHORT).show();
67. return false;
68. }
69. }

STEP 2: 设置手势识别

我们可以在Activity里设置手势识别：

1. package noodies.blog.csdn.net;
3. import android.app.Activity;
4. import android.os.Bundle;
5. import android.view.GestureDetector;
6. import android.view.MotionEvent;
8. public class GestureTestActivity extends Activity {
9. private GestureDetector mGestureDetector;
11. @Override
12. public void onCreate(Bundle savedInstanceState) {
13. super.onCreate(savedInstanceState);
14. setContentView(R.layout.main);
16. mGestureDetector = new GestureDetector(this, new MyGestureListener(this));
17. }
19. @Override
20. public boolean onTouchEvent(MotionEvent event) {
21. return mGestureDetector.onTouchEvent(event);
22. }
23. }

也可以在自定义的View里面设置手势识别：

1. package noodies.blog.csdn.net;
3. import android.content.Context;
4. import android.util.AttributeSet;
5. import android.view.GestureDetector;
6. import android.view.MotionEvent;
7. import android.view.View;
9. public class MyView extends View {
11. private GestureDetector mGestureDetector;
13. public MyView(Context context, AttributeSet attrs) {
14. super(context, attrs);
16. mGestureDetector = new GestureDetector(context, new MyGestureListener(context));
18. setLongClickable(true);
20. this.setOnTouchListener(new OnTouchListener() {
22. public boolean onTouch(View v, MotionEvent event) {
23. return mGestureDetector.onTouchEvent(event);
24. }
26. });
27. }
28. }

* 陷阱

对于自定义View，使用手势识别有两处陷阱可能会浪费你的不少时间。

1：View必须设置longClickable为true，否则手势识别无法正确工作，只会返回Down, Show, Long三种手势

2：必须在View的onTouchListener中调用手势识别，而不能像Activity一样重载onTouchEvent，否则同样手势识别无法正确工作

* 测试结果

下面是各种操作返回的手势序列，数值0表示触摸屏按下，1表示抬起

1. 单击：down 0, single up 1, single conf 0
2. 短按：down 0, show 0, single up 1
3. 长按：down 0, show 0, long 0
4. 双击：down 0, single up 1, double 0, double event 0, down 0, double event 1
5. 滚动：down 0, (show 0), scrool 2...
6. 滑动：down 0, (show 0), scrool 2..., fling 1