在Android中，有些场景需要使用代码来模拟人的操作，比如微信自动抢红包、UI测试等都需要模拟实现点击事件（Click）、触摸事件（Touch）、键盘事件（KeyBoard）等。那么，有没有现成的方案可以实现呢？答案当然是肯定的啦，往下看。

经过调研发现，给系统模拟注入输入事件有如下几种方式：

**一、使用shell命令**

使用android自带的adb shell，里面自带一个input工具，使用方法如下：

adb shell #进入系统

input keyevent KEYCODE\_BACK #模拟按返回键

input keyevent KEYCODE\_HOME #模拟按Home键

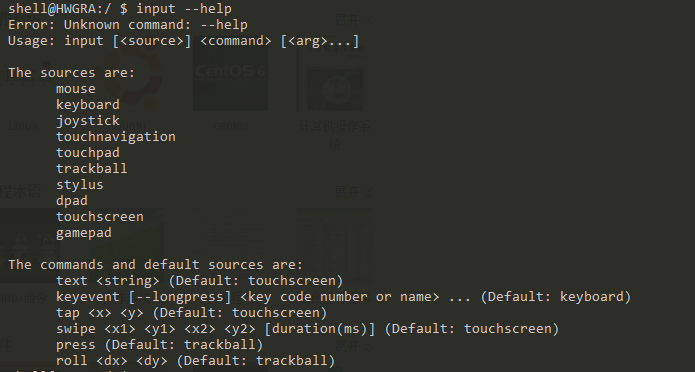
* 1
* 2
* 3

还可以直接输入点击屏幕的事件，模拟点击屏幕：

input tap 100 200 #在屏幕坐标(100, 200)处点击

* 1

详细的用法如下：



**二、使用 Instrumentation**

Instrumentation本身是Android用来做测试的工具，可以通过它监测系统与应用程序之间的交互。详情可以参考官方文档[[Test Your App]](https://developer.android.com/studio/test/index.html)。我们这里只关注怎么使用Instrumentation产生发送按键或者触屏事件。

它可以发送按键：

Instrumentation mInst = new Instrumentation();

mInst.sendKeyDownUpSync(KeyEvent.KEYCODE\_CAMERA);

* 1
* 2

也可以发送触屏事件：

Instrumentation mInst = new Instrumentation();

mInst.sendPointerSync(MotionEvent.obtain(SystemClock.uptimeMillis(),

SystemClock.uptimeMillis(), MotionEvent.ACTION\_DOWN, x, y, 0); //x,y 即是事件的坐标

mInst.sendPointerSync(MotionEvent.obtain(SystemClock.uptimeMillis(),

SystemClock.uptimeMillis(), MotionEvent.ACTION\_UP, x, y, 0);

* 1
* 2
* 3
* 4
* 5

与Shell工具一样，还有类似sendStringSync()发送文本，sendTrackballEventSync()发送轨迹球事件等方法。

sendCharacterSync(int keyCode) //用于发送指定KeyCode的按键

sendKeyDownUpSync(int key) //用于发送指定KeyCode的按键

sendPointerSync(MotionEvent event) //用于模拟Touch

sendStringSync(String text) //用于发送字符串

* 1
* 2
* 3
* 4

需要注意的是，这些方法均***不可以在UI主线程中执行***，必须放到子线程中调用，否则就会报错。另外，使用上面的方法，需要在AndroidManifast.xml中申明如下权限：

<uses-permission android:name="android.permission.INJECT\_EVENTS"/>

* 1

Demo源码下载： <https://github.com/iTimeTraveler/XYStudy>

**三、使用Android内部API**

在Android系统中，有些内部的API提供注入事件的方法。因为是内部API，在不同版本上可能变化比较大。使用如果想在普通App中使用，可能需要通过反射机制来调用。

在Android API 16之前，WindownManager有相应的方法提供注入事件的方法，如下：

IBinder wmbinder = ServiceManager.getService("window");

IWindowManager wm = IWindowManager.Stub.asInterface(wmbinder); //pointer

wm.injectPointerEvent(myMotionEvent, false); //key

wm.injectKeyEvent(new KeyEvent(KeyEvent.ACTION\_DOWN, KeyEvent.KEYCODE\_A), false);

wm.injectKeyEvent(new KeyEvent(KeyEvent.ACTION\_UP, KeyEvent.KEYCODE\_A), false); //trackball

wm.injectTrackballEvent(myMotionEvent, false);

* 1
* 2
* 3
* 4
* 5
* 6

在API 15之后，引入了InputManager，把上面的哪些injectXXXEvent()方法从WindowManager中移除了。使用方法类似：

IBinder imBinder = ServiceManager.getService("input");

IInputManager im = IInputManager.Stub.asInterface(imBinder);

//inject key event

final KeyEvent keyEvent = new KeyEvent(downTime, eventTime, action,

code, repeatCount, metaState, deviceId, scancode,

flags | KeyEvent.FLAG\_FROM\_SYSTEM |KeyEvent.FLAG\_KEEP\_TOUCH\_MODE | KeyEvent.FLAG\_SOFT\_KEYBOARD,

source);

event.setSource(InputDevice.SOURCE\_ANY)

im.injectInputEvent(keyEvent, InputManager.INJECT\_INPUT\_EVENT\_MODE\_WAIT\_FOR\_FINISH);

//inject pointer event

motionEvent.setSource(InputDevice.SOURCE\_TOUCHSCREEN);

im.injectInputEvent(motionEvent, InputManager.INJECT\_INPUT\_EVENT\_MODE\_WAIT\_FOR\_FINISH);

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14

从API 16开始，InputManager就成了一个公开的类了，可以通过如下方法获得InputManager实例：

InputManager im = (InputManager) getSystemService(Context.INPUT\_SERVICE);

* 1

注意，使用injectEvent()同样需要申明**android:name=”android.permission.INJECT\_EVENTS”**权限。

**四、可以考虑使用Monkey测试框架**

这种方案就是希望能够模拟Android Monkey的测试方法，不过博主并没有来得及对这方面进行深入的研究，可以参考这篇文章[Android Monkey源码解析](http://blog.csdn.net/xiaodanpeng/article/details/9154003)

**【参考资料】：**   
1、[Android模拟产生事件](http://www.race604.com/android-inject-input-event/)   
2、[Android 模拟键盘鼠标事件（Socket+Instrumentation实现）](http://blog.csdn.net/zhou0707/article/details/7325144)   
3、[Android Monkey源码解析](https://getpocket.com/a/read/523013957)