基于WiFi-Direct和Socket的遥控器开发

Android 4.0 及以上版本已经全面支持WiFi-Direct通信，并且已经集成到了代码中。一般在settings的wifi模块下。打开wifi即默认打开WiFi-Direct。基于WiFi-Direct的通信成本低，容易操作，方便开发。关于WiFi-Direct通信的开发流程，可以参考Android官方文档：<http://developer.android.com/guide/topics/connectivity/wifip2p.html>。这里提供了非常详细的开发流程和方法，还有一些参考代码，这里不再说明。下面主要详细讲下指令传送过程。

1. 按键

按键值可以在KeyEvent.java中找到。比如上、下、左、右四个按键所对应的值是：

KeyEvent.KEYCODE\_DPAD\_UP;

KeyEvent.KEYCODE\_DPAD\_DOWN;

KeyEvent.KEYCODE\_DPAD\_LEFT;

KeyEvent.KEYCODE\_DPAD\_RIGHT;

1. 传送按键值到服务端

假设我们已经实现了4个按键。按下后传送相应的值给服务端

**public** **void** onClick(View v) {

**switch** (v.getId()) {

**case** R.id.*btn\_left*:

sendMsg("" + KeyEvent.*KEYCODE\_DPAD\_LEFT*);

**break**;

**case** R.id.*btn\_right*:

sendMsg("" + KeyEvent.*KEYCODE\_DPAD\_RIGHT*);

**break**;

**case** R.id.*btn\_up*:

sendMsg("" + KeyEvent.*KEYCODE\_DPAD\_UP*);

**break**;

**case** R.id.*btn\_down*:

sendMsg("" + KeyEvent.*KEYCODE\_DPAD\_DOWN*);

**break**;

}

//传送按键值，单独开启一个线程

**public** **void** sendMsg(**final** String msg) {

Runnable r = **new** Runnable() {

@Override

**public** **void** run() {

**try** {

Socket client = **new** Socket();

Log.*d*("baron", "Opening client socket - ");

client.bind(**null**);

//hostAddress为服务端的ip地址,跳转activity时带过来的

client.connect((**new** InetSocketAddress(hostAddress, 8999)));

Log.*d*("baron", "Client socket - " + client.isConnected());

BufferedWriter bw = **new** BufferedWriter(**new**OutputStreamWriter(client.getOutputStream()));

bw.write(msg);

bw.flush();

client.close();

bw.close();

Log.*d*("baron", msg + " send ok.");

} **catch** (Exception e) {

Log.*e*("baron", e.getMessage());

} **finally** {

}

}

};

**new** Thread(r).start();

}

如果socket连接正常的话，通过上面的方法按键值已经准确无误的传到了服务端。下面讲服务端如何处理按键值。

1. 处理按键

我们在服务端开启一个service在后台不停的运行，多线程接收按键值。

//接收按键值

ServerSocket ss = new ServerSocket(8999);

Socket s = ss.accept();

BufferedReader br = new BufferedReader(new InputStreamReader(s. getInputStream()));

String cmd = br.readLine();

//处理按键值

Instrumentation ins = new Instrumentation();

**if** (str != **null** && !str.contains(":")){

ins.sendKeyDownUpSync(Integer.*parseInt*(str));

}

只需要执行这句系统就会处理按键效果。

1. 屏幕滑动处理

屏幕滑动要捕获滑动的动作，实现OnTouchListener，覆盖方法onTouch().

**private** **float** touchMoveX = 0;

**private** **float** touchMoveY = 0;

/\* implements method from OnTouchListener \*/

@Override

**public** **boolean** onTouch(View v, MotionEvent event) {

**if** (event.getAction() == MotionEvent.*ACTION\_MOVE*) {

**if** (touchMoveX == 0) {

touchMoveX = event.getX();

touchMoveY = event.getY();

} **else** {

sendMoveMsg(touchMoveX, touchMoveY, event.getX(), event.getY());//把4个值传过去，跟按键一样传值

}

} **else** **if** (event.getAction() == MotionEvent.*ACTION\_UP*) {

touchMoveX = event.getX();

touchMoveY = event.getY();

} **else** {

touchMoveX = 0;

touchMoveY = 0;

}

**return** **false**;

}/\* end of implements OnTouchListener \*/

滑动屏幕我们会传4个值过去，分别是down的坐标值和up的坐标值。

下面看服务端处理滑动效果。x,y就是touchMoveX和touchMoveY.

X1,y1就是getX和getY()。

MotionEvent e = MotionEvent.*obtain*(SystemClock.*uptimeMillis*(),

SystemClock.*uptimeMillis*(),

MotionEvent.*ACTION\_DOWN*, x, y, 0);

ins.sendPointerSync(e);

e = MotionEvent.*obtain*(SystemClock.*uptimeMillis*(),

SystemClock.*uptimeMillis*(),

MotionEvent.*ACTION\_MOVE*, x, y, 0);

ins.sendPointerSync(e);

e = MotionEvent.*obtain*(SystemClock.*uptimeMillis*(),

SystemClock.*uptimeMillis*(),

MotionEvent.*ACTION\_MOVE*, x, y, 0);

ins.sendPointerSync(e);

e = MotionEvent.*obtain*(SystemClock.*uptimeMillis*(),

SystemClock.*uptimeMillis*(),

MotionEvent.*ACTION\_MOVE*, x1, y1, 0);

ins.sendPointerSync(e);

e = MotionEvent.*obtain*(SystemClock.*uptimeMillis*(),

SystemClock.*uptimeMillis*(),

MotionEvent.*ACTION\_MOVE*, x1, y1, 0);

ins.sendPointerSync(e);

e = MotionEvent.*obtain*(SystemClock.*uptimeMillis*(),

SystemClock.*uptimeMillis*(),

MotionEvent.*ACTION\_UP*, x1, y1, 0);

ins.sendPointerSync(e);

到此，按键和滑动效果都处理完毕。

服务端的接收程序需要添加权限：<uses-permission android:name="android.permission.INJECT\_EVENTS" />

加了这个权限后，还要在manifest标签下添加android:sharedUserId="android.uid.system"，在MK文件中修改LOCAL\_CERTIFICATE := platform。将整个工程放到packages/apps下,编译即可。