**Input 事件在view系统中处理的流程**

**1.流程**

**InputManager -->InputChannel ---->viewrootImp**

**Viewrootimp 在初始化的时候创建一个InputStage链表**

**ViewPostImeInputStage 处理 input events to the view hierarchy**

**onTouchEvent 返回false继续分发input给子view，返回true停止分发input事件**

**Activity：**

**dispatchTouchEvent**

**onTouchEvent**

**ViewGrop：**

**dispatchTouchEvent**

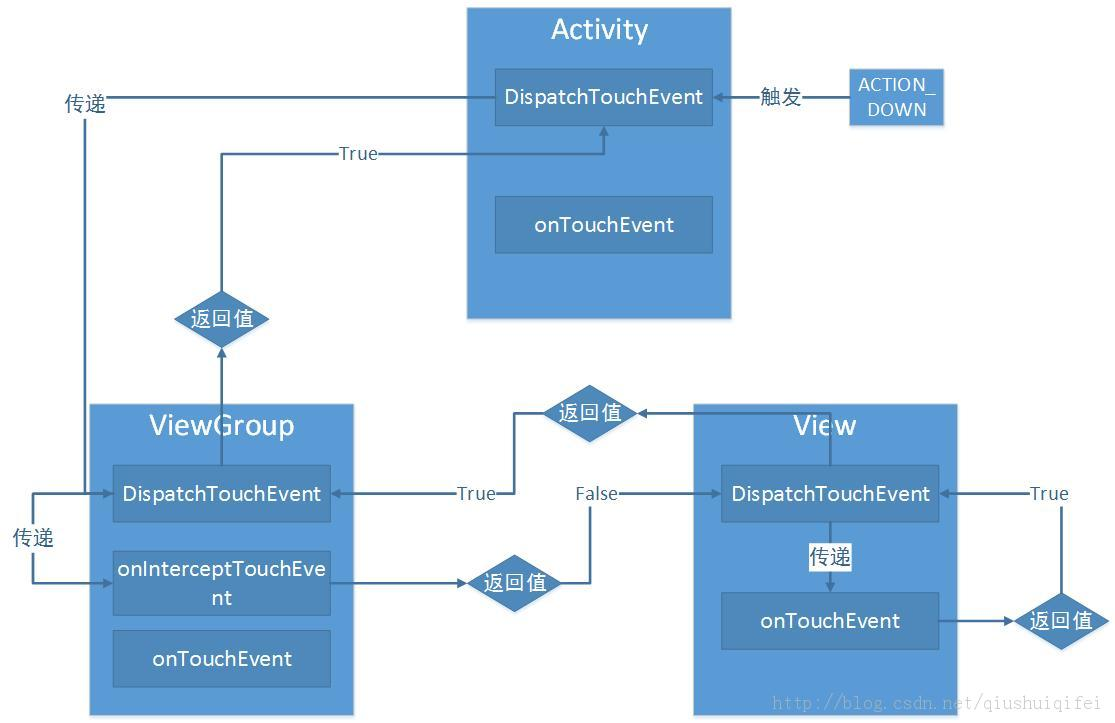
**onInterceptTouchEvent**

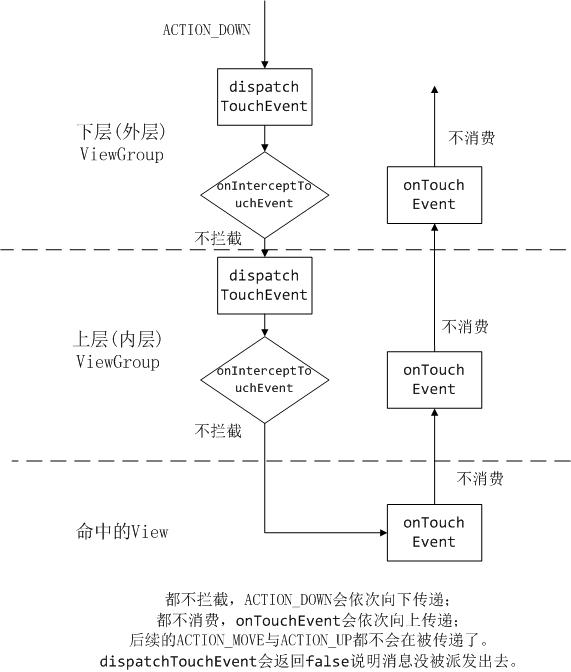
**onTouchEvent**

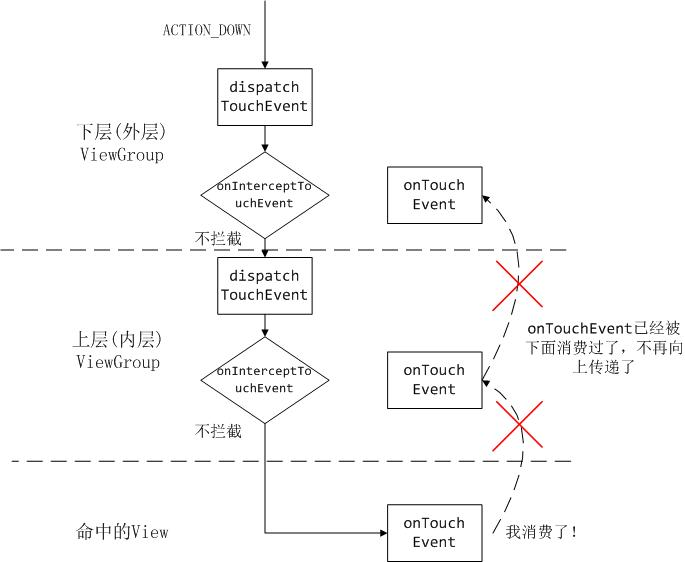
**View：**

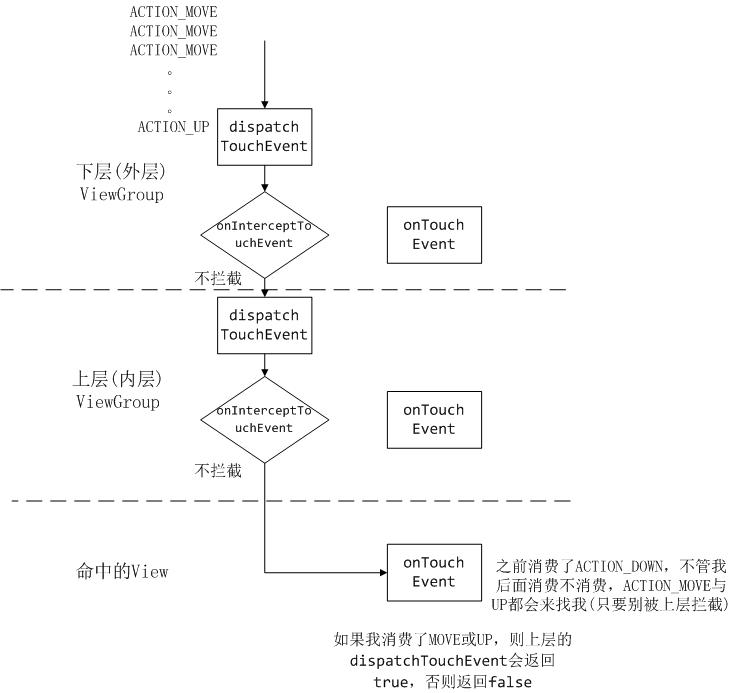
**dispatchTouchEvent**

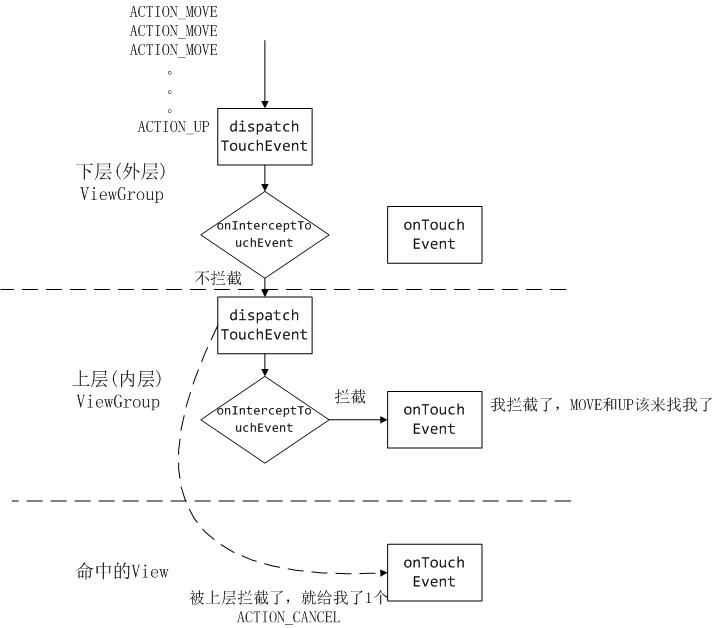
**onTouchEvent**

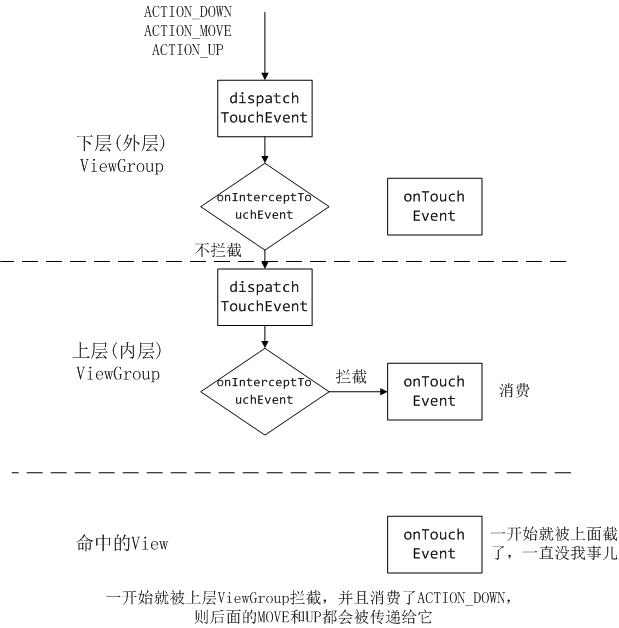












**SystemUI view的关系：**

**StatusBarWindowView**

**PhoneStatusBarView**

**NotificationPanelView**

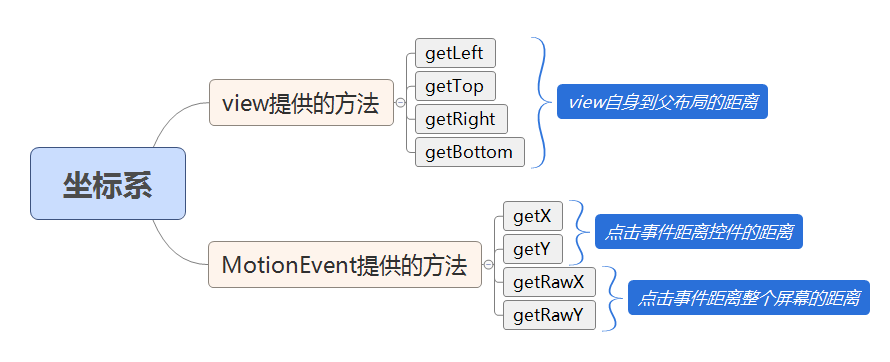
**NotificationsQuickSettingsContainer**

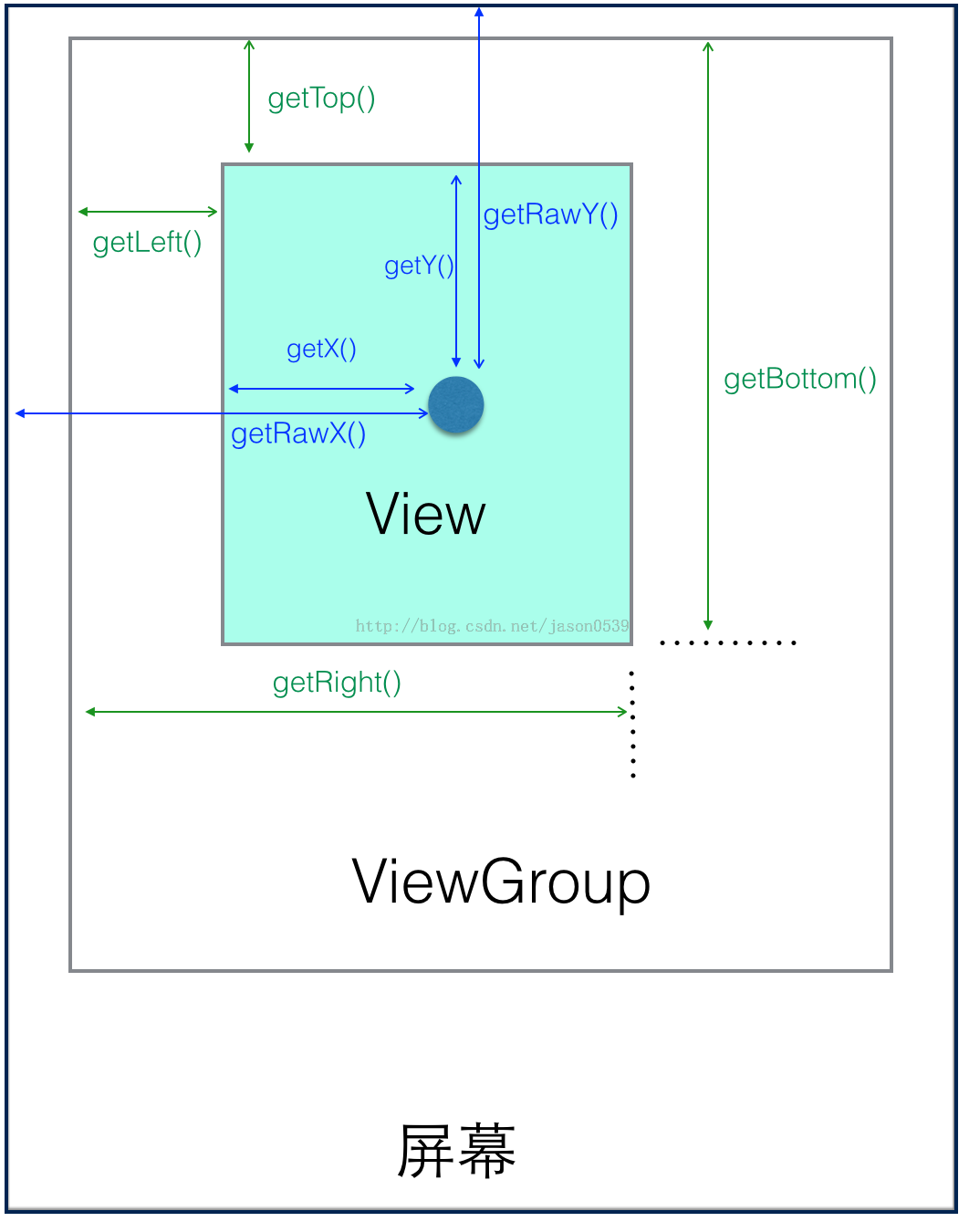
**ObservableScrollView**

**QSContainer**

**QSPanel**

**NotificationStackScrollLayout**







1. **当onInterceptTouchEvent函数拦截了一个事件，而且onTouchEvent函数返回false**

**说明没有消费相应的触摸事件，后续的move up事件也不会再传递。**

1. **当onInterceptTouchEvent函数拦截了一个事件，而且onTouchEvent函数返回true**

**Down事件会被消费，后续的move和up事件也会被派送。**

1. **onInterceptTouchEvent决定谁的onTouchEvent函数来处理event。（由上到下进行截断）**

**onTouchEvent 决定事件是否继续由下向上传递。（由下到上进行截断）**

**"main@6758" prio=5 tid=0x1 nid=NA runnable**

**java.lang.Thread.State: RUNNABLE**

**at android.view.View.dispatchTouchEvent(View.java:9963)**

**at android.view.ViewGroup.dispatchTransformedTouchEvent(ViewGroup.java:2669)**

**at android.view.ViewGroup.dispatchTouchEvent(ViewGroup.java:2344)**

**at android.view.View.dispatchPointerEvent(View.java:10254)**

**at view.ViewRootImpl$ViewPostImeInputStage.processPointerEvent(ViewRootImpl.java:4672)**

**at android.view.ViewRootImpl$ViewPostImeInputStage.onProcess(ViewRootImpl.java:4530)**

**at android.view.ViewRootImpl$InputStage.deliver(ViewRootImpl.java:4041)**

**at android.view.ViewRootImpl$InputStage.onDeliverToNext(ViewRootImpl.java:4100)**

**at android.view.ViewRootImpl$InputStage.forward(ViewRootImpl.java:4066)**

**at android.view.ViewRootImpl$AsyncInputStage.forward(ViewRootImpl.java:4215)**

**at android.view.ViewRootImpl$InputStage.apply(ViewRootImpl.java:4074)**

**at android.view.ViewRootImpl$AsyncInputStage.apply(ViewRootImpl.java:4272)**

**at android.view.ViewRootImpl$InputStage.deliver(ViewRootImpl.java:4046)**

**at android.view.ViewRootImpl$InputStage.onDeliverToNext(ViewRootImpl.java:4100)**

**at android.view.ViewRootImpl$InputStage.forward(ViewRootImpl.java:4066)**

**at android.view.ViewRootImpl$InputStage.apply(ViewRootImpl.java:4074)**

**at android.view.ViewRootImpl$InputStage.deliver(ViewRootImpl.java:4046)**

**at android.view.ViewRootImpl.deliverInputEvent(ViewRootImpl.java:6472)**

**at android.view.ViewRootImpl.doProcessInputEvents(ViewRootImpl.java:6440)**

**at android.view.ViewRootImpl.enqueueInputEvent(ViewRootImpl.java:6394)**

**At ViewRootImpl$WindowInputEventReceiver.onInputEvent(ViewRootImpl.java:6575)**

**at android.view.InputEventReceiver.dispatchInputEvent(InputEventReceiver.java:216)**

**at android.os.MessageQueue.nativePollOnce(MessageQueue.java:-1)**

**at android.os.MessageQueue.next(MessageQueue.java:323)**

**at android.os.Looper.loop(Looper.java:145)**

**at android.app.ActivityThread.main(ActivityThread.java:6485)**

**at java.lang.reflect.Method.invoke(Method.java:-1)**

**at com.android.internal.os.ZygoteInit$MethodAndArgsCaller.run(ZygoteInit.java:938)**

**at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:828)**

**分发：**

**当input事件到达ViewRootImpl之后，将该input调用根节点view 之DecorView的dispatchPointerEvent函数进行处理，由于该类没有实现dispatchPointerEvent方法，所以使用父类View的dispatchPointerEvent方法进行处理。**

**//View.java**

**//调用ViewGroup的dispatchTouchEvent函数进行input的分发**

public final boolean dispatchPointerEvent(MotionEvent event) {

if (event.isTouchEvent()) {

if (event.getAction() == MotionEvent.ACTION\_DOWN && mHasPostUnsetPressedState) {

if (getHandler() != null && getHandler().hasCallbacks(mUnsetPressedState)) {

mUnsetPressedState.run();

mHasRunUnsetPressedState = true;

}

}

return dispatchTouchEvent(event);

} else {

return dispatchGenericMotionEvent(event);

}

}

**//ViewGroup.java**

**//调用ViewGroup的dispatchTouchEvent函数进行input的分发**

@Override

public boolean dispatchTouchEvent(MotionEvent ev) {

boolean handled = false;

if (onFilterTouchEventForSecurity(ev)) {

if (actionMasked == MotionEvent.ACTION\_DOWN

|| mFirstTouchTarget != null) {

**//检查当前view是否截断了motionevent**

intercepted = onInterceptTouchEvent(ev);

} else {

// There are no touch targets and this action is not an initial down

// so this view group continues to intercept touches.

intercepted = true;

}

**//被截断就不会再去递归遍历子view**

if (!canceled && !intercepted) {

**//遍历child view树**

for (int i = childrenCount - 1; i >= 0; i--) {

final int childIndex = getAndVerifyPreorderedIndex(

childrenCount, i, customOrder);

final View child = getAndVerifyPreorderedView(

preorderedList, children, childIndex);

**//检查child view是否可见，不可见不处理input事件。**

**//检查input事件坐标是否在child view范围内。**

if (!canViewReceivePointerEvents(child)

|| !isTransformedTouchPointInView(x, y, child, null)) {

ev.setTargetAccessibilityFocus(false);

continue;

}

if (dispatchTransformedTouchEvent(ev, false, child, idBitsToAssign)) {

。。。。。。

}

}

**//当事件被截断dispatchTransformedTouchEvent child == null调用viewGroup父view**

**//的dispatchTouchEvent处理input event**

**//handled**

if (mFirstTouchTarget == null) {

// No touch targets so treat this as an ordinary view.

handled = dispatchTransformedTouchEvent(ev, canceled, null,

TouchTarget.ALL\_POINTER\_IDS);

}

**//ViewGroup.java**

**//调用ViewGroup的dispatchTransformedTouchEvent函数进行input的分发**

private boolean dispatchTransformedTouchEvent(MotionEvent event, boolean cancel,

View child, int desiredPointerIdBits) {

**//遍历到树叶端**

if (child == null) {

handled = super.dispatchTouchEvent(transformedEvent);

} else {

final float offsetX = mScrollX - child.mLeft;

final float offsetY = mScrollY - child.mTop;

transformedEvent.offsetLocation(offsetX, offsetY);

if (! child.hasIdentityMatrix()) {

transformedEvent.transform(child.getInverseMatrix());

}

**//继续递归遍历子view树**

handled = child.dispatchTouchEvent(transformedEvent);

}

}

**//View.java**

**//调用View的dispatchTouchEvent函数进行处理input event**

public boolean dispatchTouchEvent(MotionEvent event) {

if (onFilterTouchEventForSecurity(event)) {

ListenerInfo li = mListenerInfo;

if (li != null && li.mOnTouchListener != null

&& (mViewFlags & ENABLED\_MASK) == ENABLED

&& li.mOnTouchListener.onTouch(this, event)) {

result = true;

}

if (!result && onTouchEvent(event)) {

result = true;

}

}

return result;

}

**从decorView调用dispatchTouchEvent函数：**

**Down事件：**

**当该viewGroup 实现了onInterceptTouchEvent而且该函数返回true：**

**不会再去遍历该viewGroup中的child view因为mFirstTouchTarget为NULL直接调用viewGroup 父类中的dispatchTouchEvent-》onTouchEvent处理input event。**

**onInterceptTouchEvent而且该函数返回false：**

**遍历viewGroup中的child view，调用dispatchTransformedTouchEvent函数处理child view响应input event，当child view没有child view，调用child view 的onTouchEvent处理event，如果onTouchEvent返回true。父view结束对所有child view的遍历，通过addTouchTarget函数赋值mFirstTouchTarget。在递归返回的过程中创建mFirstTouchTarget链表。**

**mFirstTouchTarget链表是指哪个view处理了input event从该view到root view之间的view组成的链表。**

**onInterceptTouchEvent而且该函数返回false：**

**mFirstTouchTarget没有成功赋值，也就是所有child view没有处理input，调用当前**

**viewGroup 父类中的dispatchTouchEvent-》onTouchEvent处理input event。**

**Up,move 事件：**

**直接遍历mFirstTouchTarget链表，查找之前接受ACTION\_DOWN的孩子，并将触摸事件分配给这些孩子。**

**onInterceptTouchEvent是从上往下调用检查，onTouchEvent是从下往上调用检查。**

**参考：<http://wangkuiwu.github.io/2015/01/04/TouchEvent-ViewGroup/>**