# 8.1 原始指针事件处理

本节先来介绍一下原始指针事件(Pointer Event，在移动设备上通常为触摸事件)，下一节再介绍手势处理。

在移动端，各个平台或UI系统的原始指针事件模型基本都是一致，即：一次完整的事件分为三个阶段：手指按下、手指移动、和手指抬起，而更高级别的手势（如点击、双击、拖动等）都是基于这些原始事件的。

当指针按下时，Flutter会对应用程序执行**命中测试(Hit Test)**，以确定指针与屏幕接触的位置存在哪些组件（widget）， 指针按下事件（以及该指针的后续事件）然后被分发到由命中测试发现的最内部的组件，然后从那里开始，事件会在组件树中向上冒泡，这些事件会从最内部的组件被分发到组件树根的路径上的所有组件，这和Web开发中浏览器的**事件冒泡**机制相似， 但是Flutter中没有机制取消或停止“冒泡”过程，而浏览器的冒泡是可以停止的。注意，只有通过命中测试的组件才能触发事件。

Flutter中可以使用Listener来监听原始触摸事件，按照本书对组件的分类，则Listener也是一个功能性组件。下面是Listener的构造函数定义：

Listener({  
 Key key,  
 this.onPointerDown, //手指按下回调  
 this.onPointerMove, //手指移动回调  
 this.onPointerUp,//手指抬起回调  
 this.onPointerCancel,//触摸事件取消回调  
 this.behavior = HitTestBehavior.deferToChild, //在命中测试期间如何表现  
 Widget child  
})

我们先看一个示例，后面再单独讨论一下behavior属性。

...  
//定义一个状态，保存当前指针位置  
PointerEvent \_event;  
...  
Listener(  
 child: Container(  
 alignment: Alignment.center,  
 color: Colors.blue,  
 width: 300.0,  
 height: 150.0,  
 child: Text(\_event?.toString()??"",style: TextStyle(color: Colors.white)),  
 ),  
 onPointerDown: (PointerDownEvent event) => setState(()=>\_event=event),  
 onPointerMove: (PointerMoveEvent event) => setState(()=>\_event=event),  
 onPointerUp: (PointerUpEvent event) => setState(()=>\_event=event),  
),

运行后效果如图8-1所示：

图8-1

手指在蓝色矩形区域内移动即可看到当前指针偏移，当触发指针事件时，参数PointerDownEvent、PointerMoveEvent、PointerUpEvent都是PointerEvent的一个子类，PointerEvent类中包括当前指针的一些信息，如：

* position：它是鼠标相对于当对于全局坐标的偏移。
* delta：两次指针移动事件（PointerMoveEvent）的距离。
* pressure：按压力度，如果手机屏幕支持压力传感器(如iPhone的3D Touch)，此属性会更有意义，如果手机不支持，则始终为1。
* orientation：指针移动方向，是一个角度值。

上面只是PointerEvent一些常用属性，除了这些它还有很多属性，读者可以查看API文档。

现在，我们重点来介绍一下behavior属性，它决定子组件如何响应命中测试，它的值类型为HitTestBehavior，这是一个枚举类，有三个枚举值：

* deferToChild：子组件会一个接一个的进行命中测试，如果子组件中有测试通过的，则当前组件通过，这就意味着，如果指针事件作用于子组件上时，其父级组件也肯定可以收到该事件。
* opaque：在命中测试时，将当前组件当成不透明处理(即使本身是透明的)，最终的效果相当于当前Widget的整个区域都是点击区域。举个例子：
* Listener(  
   child: ConstrainedBox(  
   constraints: BoxConstraints.tight(Size(300.0, 150.0)),  
   child: Center(child: Text("Box A")),  
   ),  
   //behavior: HitTestBehavior.opaque,  
   onPointerDown: (event) => print("down A")  
  ),
* 上例中，只有点击文本内容区域才会触发点击事件，因为 deferToChild 会去子组件判断是否命中测试，而该例中子组件就是 Text("Box A") 。 如果我们想让整个300×150的矩形区域都能点击我们可以将behavior设为HitTestBehavior.opaque。注意，该属性并不能用于在组件树中拦截（忽略）事件，它只是决定命中测试时的组件大小。
* translucent：当点击组件透明区域时，可以对自身边界内及底部可视区域都进行命中测试，这意味着点击顶部组件透明区域时，顶部组件和底部组件都可以接收到事件，例如：
* Stack(  
   children: <Widget>[  
   Listener(  
   child: ConstrainedBox(  
   constraints: BoxConstraints.tight(Size(300.0, 200.0)),  
   child: DecoratedBox(  
   decoration: BoxDecoration(color: Colors.blue)),  
   ),  
   onPointerDown: (event) => print("down0"),  
   ),  
   Listener(  
   child: ConstrainedBox(  
   constraints: BoxConstraints.tight(Size(200.0, 100.0)),  
   child: Center(child: Text("左上角200\*100范围内非文本区域点击")),  
   ),  
   onPointerDown: (event) => print("down1"),  
   //behavior: HitTestBehavior.translucent, //放开此行注释后可以"点透"  
   )  
   ],  
  )
* 上例中，当注释掉最后一行代码后，在左上角200\*100范围内非文本区域点击时（顶部组件透明区域），控制台只会打印“down0”，也就是说顶部组件没有接收到事件，而只有底部接收到了。当放开注释后，再点击时顶部和底部都会接收到事件，此时会打印：
* I/flutter ( 3039): down1  
  I/flutter ( 3039): down0
* 如果behavior值改为HitTestBehavior.opaque，则只会打印“down1”。

### 忽略PointerEvent

假如我们不想让某个子树响应PointerEvent的话，我们可以使用IgnorePointer和AbsorbPointer，这两个组件都能阻止子树接收指针事件，不同之处在于AbsorbPointer本身会参与命中测试，而IgnorePointer本身不会参与，这就意味着AbsorbPointer本身是可以接收指针事件的(但其子树不行)，而IgnorePointer不可以。一个简单的例子如下：

Listener(  
 child: AbsorbPointer(  
 child: Listener(  
 child: Container(  
 color: Colors.red,  
 width: 200.0,  
 height: 100.0,  
 ),  
 onPointerDown: (event)=>print("in"),  
 ),  
 ),  
 onPointerDown: (event)=>print("up"),  
)

点击Container时，由于它在AbsorbPointer的子树上，所以不会响应指针事件，所以日志不会输出“in”，但AbsorbPointer本身是可以接收指针事件的，所以会输出“up”。如果将AbsorbPointer换成IgnorePointer，那么两个都不会输出。