HenCoder Plus 讲义

双向滑动的 ScalableImageView

GestrueDetector

用于在点击和长按之外，增加其他手势的监听，例如双击、滑动。通过在 `View.onTouchEvent()` 里调用 `GestureDetector.onTouchEvent()`，以代理的形式来实现：

```
override fun onTouchEvent(event: MotionEvent): Boolean {
  return gestureDetector.onTouchEvent(event)
}
```

GeasureDetector 的默认监听器：OnGestureListener

通过构造方法 `GeasureDetector(Context, OnGestureListener)` 来配置：

```
private gestureDetector = GestureDetectorCompat(context,
gestureListener)
```

OnGestureListener 的几个回调方法：

```
override fun onDown(e: MotionEvent): Boolean {
  // 每次 ACTION_DOWN 事件出现的时候都会被调用，在这里返回 true
可以保证必然消费掉事件
  return true
}
```

```
override fun onShowPress(e: MotionEvent) {
  // 用户按下 100ms 不松手后会被调用，用于标记「可以显示按下状态了」
}

override fun onSingleTapUp(e: MotionEvent): Boolean {
  // 用户单击时被调用(支持长按时长按后松手不会调用、双击的第二下时不会被调用)
  return false
}

override fun onScroll(downEvent: MotionEvent, currentEvent: MotionEvent, distanceX: Float, distanceY: Float): Boolean {
  // 用户滑动时被调用
  // 第一个事件是用户按下时的 ACTION_DOWN 事件，第二个事件是当前事件
  // 偏移是按下时的位置 - 当前事件的位置
  return false
}

override fun onLongPress(e: MotionEvent) {
  // 用户长按（按下 500ms 不松手）后会被调用
  // 这个 500ms 在 GestureDetectorCompat 中变成了 600ms (？？？)
}

override fun onFling(downEvent: MotionEvent, currentEvent: MotionEvent, velocityX: Float, velocityY: Float): Boolean {
  // 用于滑动时迅速抬起时被调用，用于用户希望控件进行惯性滑动的场景
  return false
}
```

双击监听器：OnDoubleTapListener

通过
`GestureDetector.setOnDoubleTapListener(OnDoubleTapListener)`
来配置：

```
gestureDetector.setOnDoubleTapListener(doubleTapListener
);
```

OnDoubleTapListener 的几个回调方法：

```
override fun onSingleTapConfirmed(e: MotionEvent):
Boolean {
  // 用户单击时被调用
  // 和 onSingltTapUp() 的区别在于，用户的一次点击不会立即调用这个方法，而是在一定时间后（300ms），确认用户没有进行双击，这个方法才会被调用
  return false
}

override fun onDoubleTap(e: MotionEvent): Boolean {
  // 用户双击时被调用
  // 注意：第二次触摸到屏幕时就调用，而不是抬起时
  return false
}

override fun onDoubleTapEvent(e: MotionEvent): Boolean {
  // 用户双击第二次按下时、第二次按下后移动时、第二次按下后抬起时都会被调用
  // 常用于「双击拖拽」的场景
  return false
}
```

OverScroller

用于自动计算滑动的偏移。

```
scroller = OverScroller(context);
```

常用于 `onFling()` 方法中，调用 `OverScroller.fling()` 方法来启动惯性滑动的计算：

```
override fun onFling(downEvent: MotionEvent,
currentEvent: MotionEvent, velocityX: Float, velocityY:
Float) {
  // 初始化滑动
  scroller.fling(startX, startY, velocityX, velocityY,
minX, maxX, minY, maxY)
  // 下一帧刷新
  ViewCompat.postOnAnimation(this, this)
  return false
}

...

override fun run() {
  // 计算此时的位置，并且如果滑动已经结束，就停止
  if (scroller.computeScrollOffset()) {
    // 把此时的位置应用于界面
    offsetX = scroller.currX.toFloat()
    offsetY = scroller.currY.toFloat()
    invalidate()
    // 下一帧刷新
    ViewCompat.postOnAnimation(this, this)
  }
}
```

物理模型

> 动态版本见课程视频

ScaleGestureDetector 和 ScaleGestureListener

```
private scaleGestureDetector =
ScaleGestureDetector(context, scaleGestureListener)
```

```
class HenScaleGestureListener : OnScaleGestureListener {
  override fun onScaleBegin(detector:
ScaleGestureDetector): Boolean {
    // 捏撑开始
    return true
  }

  override fun onScaleEnd(detector:
ScaleGestureDetector) {
    // 捏撑结束
  }

  override fun onScale(detector: ScaleGestureDetector):
Boolean {
    // 新的捏撑事件
    currentScale *= detector.scaleFactor
    // 这个返回值表示「事件是否消耗」，即「这个事件算不算数」
    return true
  }
}
```

问题和建议?

课上技术相关的问题，都可以去群里和大家讨论，对于比较通用的、有价值的问题，可以去我们的知识星球提问。

具体技术之外的问题和建议，都可以找丢物线（微信：diuwuxian），丢丢会为你解答技术以外的一切。

觉得好?

如果你觉得课程很棒，欢迎给我们好评呀！https://ke.qq.com/comment/index.html?cid=381952

> 一定要是你真的觉得好，再给我们好评。不要仅仅因为对扔物线的支持而好评（报名课程已经是你最大的支持了，再不够的话 B 站多来点三连我也很开心），另外我们也坚决不做好评返现等任何的交易。我们只希望，在课程对你有帮助的前提下，可以看到你温暖的评价。

更多内容：

- 网站：https://hencoder.com；https://kaixue.io
- 各大搜索引擎、微信公众号、微博、知乎、掘金、哔哩哔哩、YouTube、西瓜视频、抖音、快手、微视：统一账号「**扔物线**」，我会持续输出优质的技术内容，欢迎大家关注。
- 哔哩哔哩快捷传送门：https://space.bilibili.com/27559447

大家如果喜欢我们的课程，还请去扔物线的哔哩哔哩，帮我素质三连，感谢大家！