上一篇《[Qt Quick事件处理之鼠标、键盘、定时器](http://blog.csdn.net/foruok/article/details/30474299" \t "_blank)》我们介绍了常见的鼠标、键盘、定时器的处理，鼠标、键盘都是电脑上我们最常使用的事件，这节我们来介绍 Android 智能手机上的一个非常重要的手势：捏拉手势。捏拉手势最早在苹果手机上得到应用，苹果还曾经尝试为此操作申请专利，借以钳制三星在美国的手机和平板销售。这些我们且不管它，咱们只说在 Qt Quick 中如何处理捏拉手势。

    Qt Quick 中的 PinchArea 带来捏拉手势，看名字是不是和 MouseArea 类似？木错，就是酱紫。先来看 PinchArea 都有哪些属性和信号，了解了这些才能使用它。

**PinchArea简介**

    PinchArea 本身是一个不可见的 Item ，通常和一个可见的 Item 配合使用来处理捏拉手势。比如和一个 Rectangle 一块使用，或者和 Image 一块使用实现图片缩放、旋转等效果。捏拉手势识别实际上是通过对若干触摸事件的处理来实现的。

**属性**

    PinchArea 是 Item 的派生类，除了继承自 Item 的属性外，还有两个专属属性： enabled 和 pinch 。

    enabled 属性默认值为 true ，如果你设置为 false ，那么 PinchArea 就嘛事儿不干，捏拉区域对鼠标、触摸事件就变透明了。

    pinch 属性知名与捏拉手势的详情，它是一个组合属性，包括 target 、 active 、 minimumScale 、 maximumScale 、 minimumRotation 、 maximumRotation 、 dragAxis 、 minimumX 、 maximumX 、 minimumY 、 maximumY 等属性。

    target 指明捏拉手势要操作的 Item ， active （bool类型）属性表示目标 Item 是否正在被拖动。

    minimumScale / maximumScale 设置最小、最大缩放系数。 minimuxRotation / maximumRotation 设置对小、最大旋转角度。这四个属性都是 real 类型。

    dragAxis 设置沿 X 轴（Pinch.XAxis） 、 Y 轴（Pinch.YAxis） 还是 XY （Pinch.XAndYAxis）两个轴拖动，你也可以禁止拖动，只要给 dragAxis 赋值 Pinch.NoDrag 即可。当 dragAxis 允许拖动时，minimumX / maximumX 设定 X 轴的最小、最大拖动位置， minimumY / maximumY 设定 Y 轴的最小、最大拖动位置。

**信号**

    PinchArea 有三个信号：onPinchStarted() 、 onPinchUpdated() 、 onPinchFinished() 。它们都有一个名为 pinch 的参数，类型是 PinchEvent 。为了有效响应这些信号，必须了解 PinchEvent 类型，我们先介绍它。

    PinchEvent 具有下列属性：

* accepted ，在 onPinchStarted() 信号处理器中设置为 true 表明你要响应 PinchEvent ，Qt 会持续发给你更新事件；设置为 false ，Qt 就不会再发 PinchEvent 事件给你了。
* angle ，表示最近两个触点之间的角度， previousAngle 是上一次事件的角度， rotation 是从捏拉手势开始到当前事件的总的旋转角度。
* scale ，表示最近两个触点之间的缩放系数， previousScale 是上一次事件的缩放系数。
* center ，两个触点的中心点， previousCenter 是上一次事件的中心点， startCenter 是事件开始时的中心点。
* point1 ， point2 保存当前触点的位置， startPoint1 ， startPoint2 保存第二个触点按下时两个触点的位置。
* pointCount 保存到现在为止的触点总数。

    onPinchStarted() 信号在第一次识别到捏拉手势时发出，如果你要处理它，那就要将其设置为 true 。然后就可以通过 pinch 参数来设置要变换的 Item 了。

    当你在 onPinchStarted() 的信号处理器中接受了 TouchEvent 事件，那么 Qt 就会不断的发送新事件给你， onPinchUpdated() 信号就会不断的发射，你可以在它的信号处理器中通过 pinch 参数，撷取你需要的值来更新你的 Item 。

    onPinchFinished() 信号在用户手指离开屏幕时触发。

**怎样使用**

    介绍了 PinchArea 和 PinchEvent ，是时候看看怎么使用它们了。

    要想使用 PinchArea 来变换一个 Item ，有两个办法：

1. 设定 target 属性，将其指向要变换的 Item ，然后 PinchArea 就会在合适的时候帮你变换它。
2. 处理 onPinchStarted() / onPinchUpdated() / onPinchFinished() 信号，在信号处理器中变换目标 Item 。这种方式更灵活，你甚至可以同时处理多个 Item 。

    选定一种方式后，你可能还要配置 PinchArea.pinch 属性，给不同的参数设置合理的值，比方说最大可以放大到多少倍。

**缩放与旋转实例**

    这里提供一个简单的使用 PinchArea 的实例，我们用 PinchArea 来旋转和缩放一个矩形。

    项目的创建参考《[Qt Quick 之 Hello World 图文详解](http://blog.csdn.net/foruok/article/details/28850879" \t "_blank)》，这里不再赘述。

**使用 pinch.target**

    我们先用第一种方式，指定 pinch.target 。直接看 main.qml 文档吧：

**[javascript]** [view plaincopy](http://blog.csdn.net/foruok/article/details/32078761)[在CODE上查看代码片](https://code.csdn.net/snippets/397272)

1. **import** QtQuick 2.0
3. Rectangle {
4. width: 360;
5. height: 360;
6. focus: **true**;
7. Rectangle {
8. width: 100;
9. height: 100;
10. color: "blue";
11. id: transformRect;
12. anchors.centerIn: parent;
13. }
14. PinchArea {
15. anchors.fill: parent
16. pinch.maximumScale: 20;
17. pinch.minimumScale: 0.2;
18. pinch.minimumRotation: 0;
19. pinch.maximumRotation: 90;
20. pinch.target: transformRect;
21. }
22. }

    代码很简单，初始化了最小、最大缩放系数，最小、最大旋转角度，然后将 pinch.target 指向 id 为 transformRect 的蓝色矩形。于是，一切都正常运转，两指捏拉之间，缩放与旋转效果就出来了。

    图 1 是在 Android 手机上运行的起始效果：

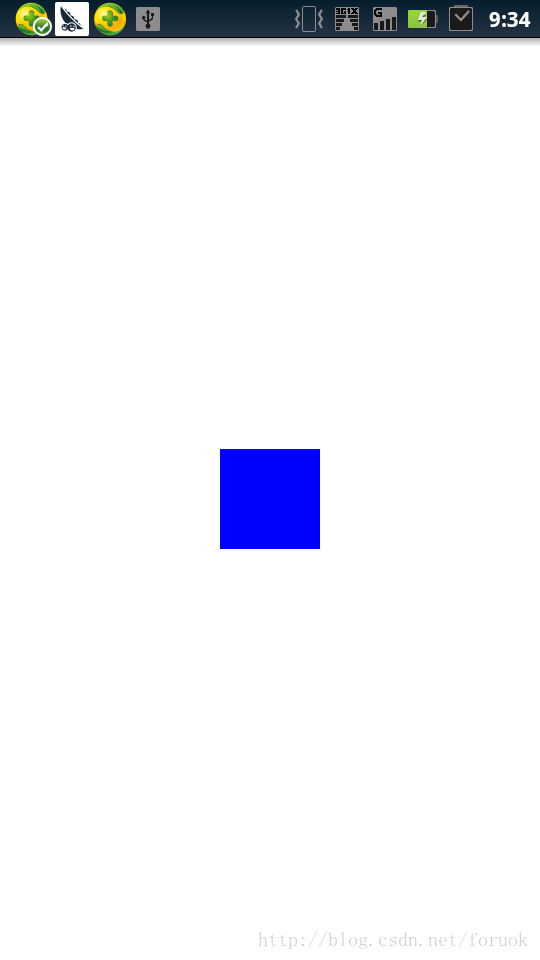


            图 1 pinchArea 示例启动效果

    图 2 是我两个指头捏拉 了几下后的效果：

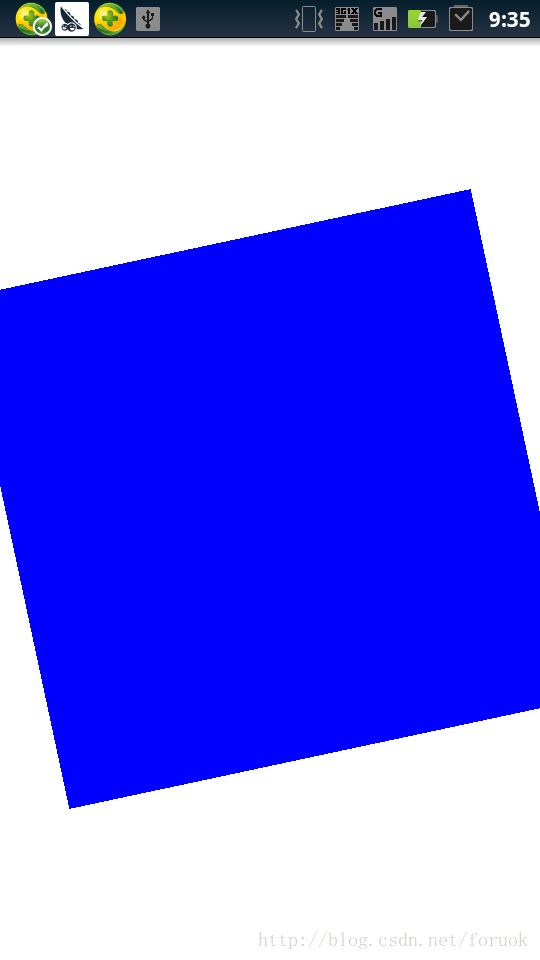


            图 2 pinchArea 缩放与旋转效果

    使用 pinch.target 这种方式，你什么都不用关心，甚至不需要弄明白 pinch 属性到底是什么含义，就可以得到一个不错的变换效果， Qt Quick 默认帮你处理所有的事情。

    下面看看使用信号的方式。

**使用信号**

    使用 onPinchStarted() / onPinchUpdated() / onPinchFinished() 要稍微麻烦一些，你必须要了解 PinchEvent 每个参数的含义，自己设计变换策略。不过好处是，七十二般变化都由你控制。

    直接看新的 main.qml 文档：

**[javascript]** [view plaincopy](http://blog.csdn.net/foruok/article/details/32078761)[在CODE上查看代码片](https://code.csdn.net/snippets/397272)

1. **import** QtQuick 2.0
3. Rectangle {
4. width: 360;
5. height: 360;
6. focus: **true**;
7. Rectangle {
8. width: 100;
9. height: 100;
10. color: "blue";
11. id: transformRect;
12. anchors.centerIn: parent;
13. }
14. PinchArea {
15. anchors.fill: parent
16. pinch.maximumScale: 20;
17. pinch.minimumScale: 0.2;
18. pinch.minimumRotation: 0;
19. pinch.maximumRotation: 90;
21. onPinchStarted: {
22. pinch.accepted = **true**;
23. }
24. onPinchUpdated: {
25. transformRect.scale \*= pinch.scale;
26. transformRect.rotation += pinch.rotation;
27. }
28. onPinchFinished: {
29. transformRect.scale \*= pinch.scale;
30. transformRect.rotation += pinch.rotation;
31. }
32. }
33. }

    代码大部分都和 pinch.target 方式一样，只是去掉了 "pinch.target: transformRect" 语句，改用信号处理器。代码很直接，不再解释了。

    OK，这篇到此结束。