**[Qt之深入理解gesture](http://blog.csdn.net/max__payne/article/details/6065681)**

Qt最新的版本4.7中有五个gesture，分别是QPanGesture, QPinchGesture, QSwipeGesture, QTapAndHoldGesture, and QTapGesture， 前三个比较常用，但用过的人仍然很少，很多人人对这个gesture意思感觉也陌生，更不用说对说出他们之间的区别了。这里我先解释一下：  
Pan Gesture就是指一个手指在屏幕上滑动，当滑动距离超过一定大小时就产生了一个Pan Gesture的事件了，如下图所示：  
  
Pinch Gesture就是两指放在屏幕上，靠近或者远离，就像捏东西一样，如下图所示：  
  
Swipe Gesture就是一个手指在屏幕上沿着一个固定的方向滑动，滑动一段距离时就产生了一个Swipe Gesture的事件，如下图所示：  
  
QTapAndHoldGesture和QTapGesture比较简单这里我就偷个懒不说了，哈哈。  
Gesture是个手势，但是个组合拳，过滤Gesture是消耗很大的，一般来说它要过滤好几个事件，要与你要截获的Gesture对比，看有没有可能是你要的那个手势，如果所有事件都会做很多无用功，因此只有真的我们需要截获那个gesture时才会让系统帮我们这么做。  
  
我比较喜欢看源码说事，就来看看  
每个gesture都有对应的gesture recognizer(手势识别类)来识别这个gesture，比如QPanGesture，它就有QPanGestureRecognizer。可有了 gesture recognizer，系统就认识你这个gesture了吗？不是，你还要将你的gesture recognizer注册到QGestureManager中，而QGestureManager则统一管理所有的gesture recognizer，系统只要通过QGestureManager就可以来识别相应的gesture。让我们来看看QGestureManager的构造函数源码：

1. QGestureManager::QGestureManager(QObject \*parent )
2. : QObject(parent), state(NotGesture), m\_lastCustomGestureId(0)
3. {
4. qRegisterMetaType<Qt::GestureState>();
5. #if defined(Q\_WS\_MAC)
6. registerGestureRecognizer(new QMacSwipeGestureRecognizer);
7. registerGestureRecognizer(new QMacPinchGestureRecognizer);
8. #if defined(QT\_MAC\_USE\_COCOA)
9. registerGestureRecognizer(new QMacPanGestureRecognizer);
10. #endif
11. #else
12. registerGestureRecognizer(new QPanGestureRecognizer);
13. registerGestureRecognizer(new QPinchGestureRecognizer);
14. registerGestureRecognizer(new QSwipeGestureRecognizer);
15. registerGestureRecognizer(new QTapGestureRecognizer);
16. #endif
17. #if defined(Q\_OS\_WIN)
18. #if !defined(QT\_NO\_NATIVE\_GESTURES)
19. if (QApplicationPrivate::HasTouchSupport)
20. registerGestureRecognizer(new QWinNativePanGestureRecognizer);
21. #endif
22. #else
23. registerGestureRecognizer(new QTapAndHoldGestureRecognizer);
24. #endif
25. }

*复制代码*

上面的代码很简单这里我只简单说说，registerGestureRecognizer就是用来注册gesture recognizer的，刚开始说的五个gesture在这里都注册了。  
我们再了看看registerGestureRecognizer函数：

1. Qt::GestureType QGestureManager::registerGestureRecognizer(QGestureRecognizer \*recognizer)
2. {
3. //这里也用你的recognizer创建一个你的gesture实例，如果没有创建成功就
4. //返回Qt::GestureType(0)，表示注册失败
5. QGesture \*dummy = recognizer->create(0);
6. if (!dummy) {
7. qWarning("QGestureManager::registerGestureRecognizer: "
8. "the recognizer fails to create a gesture object, skipping registration.");
9. return Qt::GestureType(0);
10. }
11. //Code A
12. //这里如果发现你的gestureType是CustomGesture则会
13. //返回Qt::GestureType(Qt::CustomGesture + m\_lastCustomGestureId)
14. //即在Qt::CustomGesture基础上加一个数字，如果是注册第一个的自定义的gesture，
15. //这里的m\_lastCustomGestureId就是1，再注册就加2，以此类推
16. //这里对我们理解自定义gesture很有帮助:)
17. Qt::GestureType type = dummy->gestureType();
18. if (type == Qt::CustomGesture) {
19. // generate a new custom gesture id
20. ++m\_lastCustomGestureId;
21. type = Qt::GestureType(Qt::CustomGesture + m\_lastCustomGestureId);
22. }
23. m\_recognizers.insertMulti(type, recognizer);
24. delete dummy;
25. return type;
26. }

*复制代码*

上面代码中注释的应该比较全，这里我只就Code A部分说一下，如果我自定义一个gesture时，无需管它的gesture类型，也不用重新实现它的gestureType方法，默认的就是 Qt::CustomGesture，这里的你的gesture类型id你可以通过调用 QGestureRecognizer::registerRecognizer方法的返回值来获得，也可以根据上面的代码在 Qt::CustomGesture基础上加上相应的数字来得到。  
  
说完这些，让我们来看看有关gesture的事件传递的过程：  
当你通过调用QWidget ::grabGesture来截获相应的gesture手势时  
如果你想截获某个手势，你需要调用QWidget::grabGesture方法来截获相应的手势类型，我们来看看QWidget::grabGesture方法源码：

1. void QWidget::grabGesture(Qt::GestureType gesture, Qt::GestureFlags flags)
2. {
3. Q\_D(QWidget);
4. //这里先向gestureContext，即手势上下文，插入这个gesture类型
5. //gestureContext是属于这个widget的QWidgetPrivate的，可见这里的截获事件紧限于这个widget
6. d->gestureContext.insert(gesture, flags);
7. // 创建gesture manager，这里用的是单例模式，哈哈
8. (void)QGestureManager::instance();
9. }

*复制代码*

这样之后，当事件最终通过经过QApplication::notify方法处理时，如果发现接受事件的widet有截获gesture时，就会将事件叫给gesture manager来过滤。  
整个过程事件传递如下图所示：