Custom Swipe Gestures in Qt

This article demonstrates how to create and use custom swipe gestures in Qt. The example demonstrates a custom swipe gesture recogniser - which is particularly useful because the default swipe recogniser doesn't work on Symbian in Qt 4.7.2 (QTBUG-14895).

Overview

The standard Qt documentation contains a good description of the default gestures are and how to use them (see Gestures Programming).

Unfortunately the default recognisers don't all work properly on Symbian devices in Qt 4.7.2 - as discussed in QTBUG-14895 the pan gesture does not work properly and the swipe gesture does not work at all (this is due to a framework problem which is still being investigated at time of writing).

The solution is to define a custom gesture recognition class, which can detect and simulate swipe gestures.

The general concept

The code below shows how to catch default swipe gesture events in your application (note that this code does not work in Qt 4.7.2 due to the previously mentioned bug in the default gesture recogniser)

- Tell the framework that you are interested in a specific gesture in your widget

```
MyWidget::MyWidget(QWidget parent): QWidget(parent)
{
   ...
   grabGesture(Qt::SwipeGesture);
   ...
}
```

- Then in the event handling code, process the gesture the way you like:

```
// virtual
bool
MyWidget::event(QEvent* pEvent)
{
   if (pEvent->type() == QEvent::Gesture) {
      return OnGestureEvent(static_cast<QGestureEvent*>(pEvent));
   }

   return parent::event(pEvent);
}

bool
MyWidget::OnGestureEvent(QGestureEvent* pEvent)
{
   QGesture *pSwipe = pEvent->gesture(Qt::SwipeGesture);
   if (pSwipe != NULL) {
      return OnSwipeGesture(static_cast<QSwipeGesture*>(pSwipe));
   } else {
      qDebug("Unexpected gesture detected. We only grab Qt::SwipeGesture ");
      return QWidget::event(pEvent);
   }
}

bool
MyWidget::OnSwipeGesture(QSwipeGesture* pSwipe)
{
   ...
   // Do something as result of the gesture
   ...
}
```

How to create the custom swipe recogniser

The code below shows how you define a custom gesture recognition class which can detect and simulate swipe gestures:

```
#include <QGestureRecognizer>

class SwipeGestureRecognizer : public QGestureRecognizer
{
private:
   static const int MINIMUM_DISTANCE = 10;

   typedef QGestureRecognizer parent;

   bool IsValidMove(int dx, int dy);

   qreal ComputeAngle(int dx, int dy);

   virtual QGesture* create(QObject* pTarget);

   virtual QGestureRecognizer::Result recognize(QGesture* pGesture, QObject *pWatched, QEvent *pEvent);

   void reset (QGesture *pGesture);
};
```

The backing code looks like this:

(C) Copyright Nokia 2013. All rights reserved.

```
#include <QGesture>
#include <QMouseEvent>
#include <math.h>
#include "SwipeGestureRecognizer.h"

bool
SwipeGestureRecognizer::IsValidMove(int dx, int dy)
{
   // The moved distance is to small to count as not just a glitch.
   if ((qAbs(dx) < MINIMUM_DISTANCE) && (qAbs(dy) < MINIMUM_DISTANCE)) {
      return false;
   }

   return true;
}


// virtual
QGesture*
SwipeGestureRecognizer::create(QObject* pTarget)
{
   qDebug("SwipeGestureRecognizer::create() called");
   QGesture *pGesture = new QSwipeGesture(pTarget);
   return pGesture;
}


// virtual
QGestureRecognizer::Result
SwipeGestureRecognizer::recognize(QGesture* pGesture, QObject *pWatched, QEvent *pEvent)
{
   QGestureRecognizer::Result result = QGestureRecognizer::Ignore;
   QSwipeGesture *pSwipe = static_cast<QSwipeGesture*>(pGesture);

   switch(pEvent->type()) {
      case QEvent::MouseButtonPress: {
         QMouseEvent* pMouseEvent = static_cast<QMouseEvent*>(pEvent);
         pSwipe->setProperty("startPoint", pMouseEvent->posF());
         result = QGestureRecognizer::MayBeGesture;
         qDebug("Swipe gesture started");
      }
      break;
      case QEvent::MouseButtonRelease: {
         QMouseEvent* pMouseEvent = static_cast<QMouseEvent*>(pEvent);
         const QVariant& propValue = pSwipe->property("startPoint");
         QPointF startPoint = propValue.toPointF();
         QPointF endPoint = pMouseEvent->posF();

         // process distance and direction
         int dx = endPoint.x() - startPoint.x();
         int dy = endPoint.y() - startPoint.y();

         if (!IsValidMove(dx, dy)) {
            // Just a click, so no gesture.
            result = QGestureRecognizer::CancelGesture;
            qDebug("Swipe gesture canceled");
         } else {
            // Compute the angle.
            qreal angle = ComputeAngle(dx, dy);
            pSwipe->setSwipeAngle(angle);
            result = QGestureRecognizer::FinishGesture;
            qDebug("Swipe gesture finished");
         }
      }
      break;
      default:
        break;
   }

   return result;
}

void
SwipeGestureRecognizer::reset(QGesture *pGesture)
{
   pGesture->setProperty("startPoint", QVariant(QVariant::Invalid));
   parent::reset(pGesture);
}

qreal
SwipeGestureRecognizer::ComputeAngle(int dx, int dy)
{
   double PI = 3.14159265;

   // Need to convert from screen coordinates direction
   // into classical coordinates direction.
   dy = -dy;

   double result = atan2((double)dy, (double)dx) ;
   result = (result * 180) / PI;

   // Always return positive angle.
   if (result < 0) {
      result += 360;
   }

   return result;
}
```

Using the custom gesture recognizer

- Create the recognizer

```
MyWidget::MyWidget(QWidget *parent) : QWidget(parent)
{
   ...
   grabGesture(Qt::SwipeGesture);

   ...

   // Create a SWIPE recognizer because the default SWIPE recognizer
   // does not really work on Symbian device.
   QGestureRecognizer* pRecognizer = new SwipeGestureRecognizer();
```


(C) Copyright Nokia 2013. All rights reserved.

```
   m_gestureId = QGestureRecognizer::registerRecognizer(pRecognizer);
}
```

- Do not forget to add appropriate removal code

```
MyWidget::~MyWidget()
{
   QGestureRecognizer::unregisterRecognizer(m_gestureId);
}
```

- Here is how the handler from the general skeleton above can look:

```
bool
MyWidget::OnSwipeGesture(QSwipeGesture* pSwipe)
{
   bool result = false;

   if (pSwipe->state() == Qt::GestureFinished) {
      qDebug("Swipe angle: %f", pSwipe->swipeAngle());
      switch (SwipeGestureUtil::GetHorizontalDirection(pSwipe)) {
         case QSwipeGesture::Left:
            qDebug("Swipe Left detected");
            ...
            result = true;
            break;
         case QSwipeGesture::Right:
            qDebug("Swipe Right detected");
            ...
            result = true;
            break;
         default:
            break;
      }
   }

   return result;
}
```

Swipe gesture utility class

The `SwipeGestureUtil` is an utility class which helps you make sense of the swipe direction. The standard `QSwipeGesture::horizontalDirection()` and `QSwipeGesture::horizontalDirection()` return **Left/Right/Up/Down** always when the angle is not **0/90/180/270/360** degrees. But what we want is to count for horizontal swipe only when the user moved his finger more horizontally than vertically. The util methods below help you achieve that goal.

```
QSwipeGesture::SwipeDirection
SwipeGestureUtil::GetHorizontalDirection(QSwipeGesture *pSwipeGesture)
{
   qreal angle = pSwipeGesture->swipeAngle();
   if (0 <= angle && angle <= 45) {
      return QSwipeGesture::Right;
   }

   if (135 <= angle && angle <= 225) {
      return QSwipeGesture::Left;
   }

   if (315 <= angle && angle <= 360) {
      return QSwipeGesture::Right;
   }

   return QSwipeGesture::NoDirection;
}

QSwipeGesture::SwipeDirection
SwipeGestureUtil::GetVerticalDirection(QSwipeGesture *pSwipeGesture)
{
   qreal angle = pSwipeGesture->swipeAngle();

   if (45 < angle && angle < 135) {
      return QSwipeGesture::Up;
   }

   if (225 < angle && angle < 315) {
      return QSwipeGesture::Down;
   }

   return QSwipeGesture::NoDirection;
}
```


(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.