在触摸设备上可以使用Qt的手势事件

要激活手势事件，需要执行以下操作：

第一步，为QWidget控件注册手势事件

QList<Qt::GestureType> gestures;

gestures << Qt::PanGesture;

gestures << Qt::PinchGesture;

gestures << Qt::SwipeGesture;

gestures << Qt::TapGesture;

gestures << Qt::TapAndHoldGesture;

MainWindow w;

w.grabGestures(gestures);

第二步，在QWidget控件类中过滤事件

bool MainWindow::*event*(QEvent \*event)

{

if (event->type() == QEvent::Gesture)

return gestureEvent(static\_cast<QGestureEvent\*>(event));

return QWidget::*event*(event);

}

bool MainWindow::gestureEvent(QGestureEvent \*event)

{

qCDebug(lcExample) << "gestureEvent():" << event->gestures().size();

if (QGesture \*swipe = event->gesture(Qt::SwipeGesture))

swipeTriggered(static\_cast<QSwipeGesture \*>(swipe));

else if (QGesture \*pan = event->gesture(Qt::PanGesture))

panTriggered(static\_cast<QPanGesture \*>(pan));

else if (QGesture \*tap = event->gesture(Qt::TapGesture))

tapTriggered(static\_cast<QTapGesture \*>(tap));

else if (QGesture \*tapAndHold = event->gesture(Qt::TapAndHoldGesture))

tapAndHoldTriggered(static\_cast<QTapAndHoldGesture \*>(tapAndHold));

if (QGesture \*pinch = event->gesture(Qt::PinchGesture))

pinchTriggered(static\_cast<QPinchGesture \*>(pinch));

return true;

}

//! [gesture event handler]

void MainWindow::panTriggered(QPanGesture \*gesture)

{

#ifndef QT\_NO\_CURSOR

switch (gesture->state()) {

case Qt::GestureStarted:

case Qt::GestureUpdated:

setCursor(Qt::SizeAllCursor);

break;

default:

setCursor(Qt::ArrowCursor);

}

#endif

QPointF delta = gesture->delta();

qCDebug(lcExample) << "panTriggered():" << delta;

horizontalOffset += delta.x();

verticalOffset += delta.y();

}

void MainWindow::pinchTriggered(QPinchGesture \*gesture)

{

QPinchGesture::ChangeFlags changeFlags = gesture->changeFlags();

if (changeFlags & QPinchGesture::RotationAngleChanged) {

const qreal value = gesture->property("rotationAngle").toReal();

const qreal lastValue = gesture->property("lastRotationAngle").toReal();

const qreal rotationAngleDelta = value - lastValue;

rotationAngle += rotationAngleDelta;

qCDebug(lcExample) << "pinchTriggered(): rotation by" << rotationAngleDelta << rotationAngle;

}

if (changeFlags & QPinchGesture::ScaleFactorChanged) {

qreal value = gesture->property("scaleFactor").toReal();

currentStepScaleFactor = value;

qCDebug(lcExample) << "pinchTriggered(): " << currentStepScaleFactor;

}

if (gesture->state() == Qt::GestureFinished) {

scaleFactor \*= currentStepScaleFactor;

currentStepScaleFactor = 1;

}

update();

}

//! [swipe function]

void MainWindow::swipeTriggered(QSwipeGesture \*gesture)

{

if (gesture->state() == Qt::GestureFinished) {

if (gesture->horizontalDirection() == QSwipeGesture::Left

|| gesture->verticalDirection() == QSwipeGesture::Up) {

qCDebug(lcExample) << "swipeTriggered(): swipe to previous";

} else {

qCDebug(lcExample) << "swipeTriggered(): swipe to next";

}

}

}

//! [swipe function]

void MainWindow::tapTriggered(QTapGesture \*gesture)

{

qCDebug(lcExample) << "tapTriggered():" ;

}

void MainWindow::tapAndHoldTriggered(QTapAndHoldGesture \*gesture)

{

qCDebug(lcExample) << "tapAndHoldTriggered():";

}