Qt QWidget实现手势缩放和平移

- QWidget中实现一个手势操作的功能，对图片进行放大 / 缩小 / 平移功能，并且还需要支持通过鼠标和键盘来实现该功能。其实这种功能在QGraphicsView中实现比较简单， 不过在QWidget中也能实现，本次通过QGestureEvent来捕捉手势操作，然后对图片进行缩放或者移动。

```
class QGestureEvent;
class QPanGesture;
class QPinchGesture;
class QSwipeGesture;
class CProjectionPicture;

class CProjectionPicture :public QWidget
{
 Q_OBJECT
public:
 CProjectionPicture(QWidget *parent = 0);
 voidsetPicture(QImage & image);
protected:
 // 放大/缩小
 voidwheelEvent(QWheelEvent *event) Q_DECL_OVERRIDE;
 voidmouseDoubleClickEvent(QMouseEvent *event) Q_DECL_OVERRIDE;
 voidkeyPressEvent(QKeyEvent *event) Q_DECL_OVERRIDE;
 // 平移
 voidmouseMoveEvent(QMouseEvent *event) Q_DECL_OVERRIDE;
 voidmousePressEvent(QMouseEvent *event) Q_DECL_OVERRIDE;
 voidmouseReleaseEvent(QMouseEvent *event) Q_DECL_OVERRIDE;
 boolevent(QEvent *event) Q_DECL_OVERRIDE;
 voidpaintEvent(QPaintEvent *event) Q_DECL_OVERRIDE;
 voidresizeEvent(QResizeEvent *e) Q_DECL_OVERRIDE;
public Q_SLOTS:
 voidzoomIn(); // 放大
 voidzoomOut(); // 缩小
 voidzoom(float scale); // 缩放 - scaleFactor: 缩放的比例因子
 voidtranslate(QPointF delta); // 平移
```

```
private:
 boolgestureEvent(QGestureEvent *event);
 voidpanTriggered(QPanGesture*);
 voidpinchTriggered(QPinchGesture*);
 QImage loadImage(const QString &fileName);
 QImage currentImage;
 qreal horizontalOffset;
 qreal verticalOffset;
 qreal scaleFactor;
 qreal currentStepScaleFactor;
 Qt::MouseButton m_translateButton; // 平移按钮
bool m_bMouseTranslate;
 qreal m_zoomDelta; // 缩放的增量
 QPoint m_lastMousePos; // 鼠标最后按下的位置
};
```

源文件：

```
CProjectionPicture::CProjectionPicture(QWidget *parent)
 : QWidget(parent),
 horizontalOffset(0),
 verticalOffset(0),
 scaleFactor(1),
 currentStepScaleFactor(1),
 m_translateButton(Qt::LeftButton),
 m_bMouseTranslate(false),
 m_zoomDelta(0.2),
{
this->setFocusPolicy(Qt::ClickFocus);
 grabGesture(Qt::PanGesture);
 grabGesture(Qt::PinchGesture);
 grabGesture(Qt::SwipeGesture);
}
void CProjectionPicture::setPicture(QImage &image)
{
 currentImage = image.convertToFormat(QImage::Format_RGB888);
```

```
 update();
}
bool CProjectionPicture::event(QEvent *event)
{
 if (event->type() == QEvent::Gesture)
 return gestureEvent(static_cast(event));
 return QWidget::event(event);
}
void CProjectionPicture::paintEvent(QPaintEvent*)
{
 QPainter painter(this);
 QImage image = currentImage;
 if(!image.isNull()){
 image = image.scaled(this->width()*currentStepScaleFactor * sc
aleFactor,
 this->height()*currentStepScaleFactor * scaleFactor,
 Qt::KeepAspectRatio,
 Qt::SmoothTransformation);
 }
 const qreal iw = image.width();
 const qreal ih = image.height();
 const qreal wh = height();
 const qreal ww = width();
 painter.translate(ww/2, wh/2);
 painter.translate(horizontalOffset, verticalOffset);
 painter.translate(-iw/2, -ih/2);
 painter.drawImage(0,0,image);
}
void CProjectionPicture::mouseDoubleClickEvent(QMouseEvent *)
{
 scaleFactor = 1;
 currentStepScaleFactor = 1;
 verticalOffset = 0;
 horizontalOffset = 0;
 update();
```

```
53 }
54 bool CProjectionPicture::gestureEvent(QGestureEvent *event)
55 {
56  if (QGesture *pan = event->gesture(Qt::PanGesture))
57  panTriggered(static_cast(pan));
58  if (QGesture *pinch = event->gesture(Qt::PinchGesture))
59  pinchTriggered(static_cast(pinch));
60  return true;
61 }
62 void CProjectionPicture::panTriggered(QPanGesture *gesture)
63 {
64  #ifndef QT_NO_CURSOR
65  switch (gesture->state()) {
66  case Qt::GestureStarted:
67  case Qt::GestureUpdated:
68  setCursor(Qt::SizeAllCursor);
69  break;
70  default:
71  setCursor(Qt::ArrowCursor);
72  }
73 #endif
74  QPointF delta = gesture->delta();
75  horizontalOffset += delta.x();
76  verticalOffset += delta.y();
77  update();
78 }
79 void CProjectionPicture::pinchTriggered(QPinchGesture *gesture)
80 {
81  QPinchGesture::ChangeFlags changeFlags = gesture-
>changeFlags();
82  if (changeFlags & QPinchGesture::ScaleFactorChanged) {
83  currentStepScaleFactor = gesture->totalScaleFactor();
84  }
85  if (gesture->state() == Qt::GestureFinished) {
86  scaleFactor *= currentStepScaleFactor;
```

```
 currentStepScaleFactor = 1;
 }
 update();
}
void CProjectionPicture::resizeEvent(QResizeEvent*e)
{
 update();
 QWidget::resizeEvent(e);
}
// 上/下/左/右键向各个方向移动、加/减键进行缩放
void CProjectionPicture::keyPressEvent(QKeyEvent *event)
{
 switch (event->key()) {
  qDebug() << event->key();
  case Qt::Key_Up:
  translate(QPointF(0, -5)); // 上移
  break;
  case Qt::Key_Down:
  translate(QPointF(0, 5)); // 下移
  break;
  case Qt::Key_Left:
  translate(QPointF(-5, 0)); // 左移
  break;
  case Qt::Key_Right:
  translate(QPointF(5, 0)); // 右移
  break;
  case Qt::Key_Plus: // 放大
  zoomIn();
  break;
  case Qt::Key_Minus: // 缩小
  zoomOut();
  break;
  default:
  QWidget::keyPressEvent(event);
  }
```

```
122  QWidget::keyPressEvent(event);
123 }
124 // 平移
125 void CProjectionPicture::mouseMoveEvent(QMouseEvent *event)
126 {
127  if (m_bMouseTranslate){
128  QPointF mouseDelta = event->pos() - m_lastMousePos;
129  translate(mouseDelta);
130  }
131  m_lastMousePos = event->pos();
132  QWidget::mouseMoveEvent(event);
133 }
134 void CProjectionPicture::mousePressEvent(QMouseEvent *event)
135 {
136  qDebug() << "CProjectionPicture::mousePressEvent";
137  if (event->button() == m_translateButton) {
138  m_bMouseTranslate = true;
139  m_lastMousePos = event->pos();
140  setCursor(Qt::OpenHandCursor);
141  }
142  QWidget::mousePressEvent(event);
143 }
144 void CProjectionPicture::mouseReleaseEvent(QMouseEvent *event)
145 {
146  if (event->button() == m_translateButton)
147  {
148  m_bMouseTranslate = false;
149  setCursor(Qt::ArrowCursor);
150  }
151  QWidget::mouseReleaseEvent(event);
152 }
153 // 放大/缩小
154 void CProjectionPicture::wheelEvent(QWheelEvent *event)
155 {
156  qDebug() << "CProjectionPicture::wheelEvent";
```

```
157  QPoint scrallAmount = event->angleDelta();
158  if(scrallAmount.y() > 0){
159  zoomIn();
160  }elseif(scrallAmount.y() < 0){
161  zoomOut();
162  }
163  QWidget::wheelEvent(event);
164 }
165 // 放大
166 void CProjectionPicture::zoomIn()
167 {
168  zoom(1 + m_zoomDelta);
169 }
170 // 缩小
171 void CProjectionPicture::zoomOut()
172 {
173  zoom(1 - m_zoomDelta);
174 }
175 // 缩放 - scaleFactor: 缩放的比例因子
176 void CProjectionPicture::zoom(float scale)
177 {
178  scaleFactor *= scale;
179  update();
180 }
181 // 平移
182 void CProjectionPicture::translate(QPointF delta)
183 {
184  horizontalOffset += delta.x();
185  verticalOffset += delta.y();
186  update();
187 }
```