How to drag & drop QAbstractGraphicsShapeItem in QGraphicsView

Problem

Implement drag and drop in QGraphicsView for QAbstractGraphicsShapeItem items.

Solution

Sublass QGraphicsScene and reimplement mousePressEvent, mouseMoveEvent and mouseReleaseEvent to capture all mouse events.

```
#ifndef MAIN_H
#define MAIN_H

#include <QtGui>

class DnDGraphicsScene : public QGraphicsScene
{
    Q_OBJECT
public:
    DnDGraphicsScene(QWidget *parent = 0);

protected:
    void mousePressEvent(QGraphicsSceneMouseEvent *mouseEvent);
    void mouseMoveEvent(QGraphicsSceneMouseEvent *mouseEvent);
    void mouseReleaseEvent(QGraphicsSceneMouseEvent *mouseEvent);

private:
    QAbstractGraphicsShapeItem *dragged;
    QPen selected;
    QBrush selectedBrush;
    QPen normal;
    QBrush normalBrush;
    QPointF offset;
};

#endif // MAIN_H
```

Press event detects beginning of a drag and release end of a drag. In between we move item over the scene.

```
#include <QtGui>

#include "main.h"

DnDGraphicsScene::DnDGraphicsScene(QWidget *parent)
  : QGraphicsScene(parent), dragged(0)
{
    selected = QPen(Qt::red, 3);
    selectedBrush  = QBrush(Qt::yellow);
}

void DnDGraphicsScene::mousePressEvent(QGraphicsSceneMouseEvent *mouseEvent)
{
    dragged = qgraphicsitem_cast<QAbstractGraphicsShapeItem*>(
          itemAt(mouseEvent->scenePos(), QTransform()));
    if (dragged) {
        offset = dragged->pos() - mouseEvent->scenePos();
        normal = dragged->pen();
        normalBrush = dragged->brush();
        dragged->setPen(selected);
        dragged->setBrush(selectedBrush);
    } else
        QGraphicsScene::mousePressEvent(mouseEvent);
}

void DnDGraphicsScene::mouseMoveEvent(QGraphicsSceneMouseEvent *mouseEvent)
{
    if (dragged) {
        dragged->setPos(offset + mouseEvent->scenePos());
    } else
        QGraphicsScene::mouseMoveEvent(mouseEvent);
}

void DnDGraphicsScene::mouseReleaseEvent(QGraphicsSceneMouseEvent *mouseEvent)
{
    if (dragged) {
        dragged->setPen(normal);
        dragged->setBrush(normalBrush);
        dragged = 0;
    } else
        QGraphicsScene::mouseReleaseEvent(mouseEvent);
}
```

In main function we put all things together

```
int main(int argc, char *argv[])
{
    QApplication a(argc, argv);

    QPolygonF polygon1;
    polygon1 << QPointF(0.0, 0.0) << QPointF(25.0, 0.0) << QPointF(12.5, 25.0);
    QGraphicsPolygonItem *item1 = new QGraphicsPolygonItem(polygon1);
    item1->setBrush(Qt::green);
    item1->setTransformOriginPoint(12.5, 12.5);
```


(C) Copyright Nokia 2013. All rights reserved.

```
    item1->setRotation(45.0);
    QPolygonF polygon2;
    polygon2 << QPointF(50.0, 50.0) << QPointF(75.0, 50.0) << QPointF(62.5, 75.0);
    QGraphicsPolygonItem *item2 = new QGraphicsPolygonItem(polygon2);
    item2->setBrush(Qt::blue);
    item2->setTransformOriginPoint(12.5, 12.5);
    item2->setRotation(-45.0);
    DnDGraphicsScene scene;
    scene.addItem(item1);
    scene.addItem(item2);
    QGraphicsView view(&scene);
    view.show();

    return a.exec();
}
```


(C) Copyright Nokia 2013. All rights reserved.

 (C) Copyright Nokia 2013. All rights reserved.