I am trying to implement a method to edit features in different geometry type. To start with, I found following example code from the book "BUILDING MAPPING APPLICATIONS WITH QGIS", but it always crashes when I call the method of self.feature in the function canvasMoveEvent. In function canvasPressEvent, it seems to work fine.

I tried to execute the code in a clean profile, but it still crashes. Then I tried with QGIS 2.18 and it worked there. I have attached the report below.

```python
class MovePointTool(QgsMapToolIdentify):
    def __init__(self, mapCanvas, layer):
        QgsMapToolIdentify.__init__(self, mapCanvas)
        self.setCursor(Qt.CrossCursor)
        self.layer = layer
        self.dragging = False
        self.feature = None

    def canvasPressEvent(self, event):
        found_features = self.identify(event.x(), event.y(), [self.layer], self.TopDownAll)
        if len(found_features) > 0:
            self.dragging = True
            self.feature = found_features[0].mFeature
            print(self.feature.id())
        else:
            self.dragging = False
            self.feature = None

    def canvasMoveEvent(self, event):
        if self.dragging and self.feature:
            point = self.toLayerCoordinates(self.layer, event.pos())
            geometry = QgsGeometry.fromPointXY(point)
            print(self.feature.isValid())
            print(self.feature.id())
            self.layer.changeGeometry(self.feature.id(), geometry)
            self.canvas().refresh()

    def canvasReleaseEvent(self, event):
        self.dragging = False
        self.feature = None

tool = MovePointTool(iface.mapCanvas(), iface.activeLayer())
iface.mapCanvas().setMapTool(tool)
```

2022-01-12
Crash Report:

Crash ID: 46eec519cad6876ebf2347eb73a9fc7f3df167fa

Stack Trace

QgsFeature::isValid:
PyInit__core:
PyMethodDef_RawFastCallKeywords:
PyMethodDef_RawFastCallKeywords:
PyEval_EvalFrameDefault:
PyFunction_FastCallDict:
PyMethodDef_RawFastCallDict:
PyObject_Call:
PyInit_sip:
PyInit__gui:
QgsMapCanvas::mouseReleaseEvent:
QWidget::event:
QFrame::event:
QGraphicsView::viewportEvent:
QCoreApplicationPrivate::sendThroughObjectEventFilters:
QApplicationPrivate::notify_helper:
QApplication::notify:
QgsApplication::notify:
QCoreApplication::notifyInternal2:
QApplicationPrivate::sendMouseEvent:
QSizePolicy::QSizePolicy:
QSizePolicy::QSizePolicy:
QCoreApplication::notifyInternal2:
QGuiApplicationPrivate::processMouseEvent:
QWindowSystemInterface::sendWindowSystemEvents:
QEventDispatcherWin32::processEvents:
TranslateMessageEx:
TranslateMessage:
QEventDispatcherWin32::processEvents:
qt_plugin_query_metadata:
QEventLoop::exec:
QCoreApplication::exec:
main:
BaseThreadInitThunk:
RtlUserThreadStart:

QGIS Info
QGIS Version: 3.4.1-Madeira
QGIS code revision: 383851c597
Compiled against Qt: 5.11.2
Running against Qt: 5.11.2
Compiled against GDAL: 2.3.2
Running against GDAL: 2.3.2

System Info
CPU Type: x86_64
Kernel Type: winnt
Kernel Version: 6.1.7601

History
#1 - 2018-12-07 03:56 PM - Giovanni Manghi
- Priority changed from Normal to High
- Category changed from Editing to Python bindings / sipify

#2 - 2019-01-28 03:40 PM - Hugo Mercier
Reproduced on 3.4.4
I guess this is because you are storing a reference to a QgsFeature that sometimes no longer exist when you use it.

    self.feature = found_features[0].mFeature

Copying the feature seems to fix the problem

    self.feature = QgsFeature(found_features[0].mFeature)

#3 - 2019-01-28 03:40 PM - Hugo Mercier
- Status changed from Open to Closed