QGIS Application - Bug report #13745
Snapping rounding errors with on-the-fly reprojection

2015-11-03 04:31 AM - Daan Goedkoop

Status:	Closed		
Priority:	Normal		
Assignee:	Martin Dobias		
Category:	Digitising		
Affected QGIS version:	master	**Regression?:**	No
Operating System:		**Easy fix?:**	No
Pull Request or Patch supplied:	No	**Resolution:**	
Crashes QGIS or corrupts data:	No	**Copied to github as #:**	21772

Description

The snapping function doesn't snap exactly to vertices when on-the-fly reprojection is activated.

In the attachments, images are shown with an example for reproducing this error. (1) Using the OpenLayers-plugin, the project is set to WGS84/Pseudo Mercator with on-the-fly reprojection enabled. (2) Create a new layer with a different CRS. (3) Digitise a line. (4) Enable snapping at vertices only. (5) For example, cut the line with snapping at a vertex. (6) When zooming in very far, it becomes apparent that the line was not actually cut at the vertex.

I've looked into the source code and to me it seems that the problem is the following. When digitising (whether to draw a new line or to draw a cutting path), the path is stored in memory using the coordinates of the project CRS. The snapping function converts the vertices of the target layer to the project CRS, and after finishing editing the digitised path is converted back to the target layer CRS. This back-and-forth-conversion can lead to floating-point math inaccuracies.

History

#1 - 2015-11-06 01:08 AM - Giovanni Manghi

- Status changed from Open to Feedback

I just followed all the steps and on QGIS master I cannot replicate the issue. Could you please test on master and/or attach a sample project with data?

#2 - 2015-11-06 09:29 AM - Daan Goedkoop

- File bug.tgz added

I have just tried it with QGIS Master on Linux, and the behaviour is as I originally described. I have attached a file with a sample project and a sample shape file that shows the problem (the line consisting of the segments 0 and 1 has not been cut exactly at the middle vertex; the line with the id 2 has not been snapped exactly to the right location either). You need to zoom in all the way to about 5000:1 in order to see the inaccuracies.

#3 - 2015-11-06 09:58 AM - Giovanni Manghi

- Affected QGIS version changed from 2.12.0 to master
- Status changed from Feedback to Open

Now I see. Thanks.

#4 - 2015-11-06 01:22 PM - Daan Goedkoop

It can be an annoying bug. For example, the 'split feature' tool has the feature that if you start cutting at a snapped vertex, the line is cut immediately, rather than starting to draw a cutting line. However, in case of an inaccuracy such as in this bug, this fails: the line is not cut. Probably because the cutting point ends up slightly besides the vertex.

The node editor tool doesn't seem to be affected.

I'm new to the Qgis code base, but I've tried to figure this out a bit. This seems to revolve around the function (declaration slightly reformatted to include the class etc.):

```
    /** Adds a point to the rubber band (in map coordinates) and to the capture list (in layer coordinates) */
    int QgsMapToolCapture::addVertex( const QgsPoint& point );
```

It is called, for example, in QgsMapToolSplitFeatures::cadCanvasReleaseEvent(QgsMapMouseEvent * e), as following:

addVertex(e->mapPoint());

On its turn, QgsMapMouseEvent::snapPoint has several code paths, all with QgsSnappingUtils::snapToMap, to calculate the mMapPoint that is returned in mapPoint().

Back to the function QgsMapToolCapture::addVertex. To create the capture list in layer coordinates, as promised in the comment, it uses QgsMapToolCapture::nextPoint, which contains the following line (including the comment):

layerPoint = toLayerCoordinates(vlayer, mapPoint); //transform snapped point back to layer crs

Thus, if I understand it correctly, the point is snapped (and transformed) to map coordinates and then transformed back to layer coordinates. I think this might be the problem, because floating point arithmetic doesn't guarantee that you end up with exactly the original value if you perform a calculation and then the inverse calculation.

#5 - 2015-11-07 12:04 AM - Giovanni Manghi

Daan Goedkoop wrote:

> *It can be an annoying bug. For example, the 'split feature' tool has the feature that if you start cutting at a snapped vertex, the line is cut immediately, rather than starting to draw a cutting line. However, in case of an inaccuracy such as in this bug, this fails: the line is not cut. Probably because the cutting point ends up slightly besides the vertex.*
>
> *The node editor tool doesn't seem to be affected.*
>
> *I'm new to the Qgis code base, but I've tried to figure this out a bit. This seems to revolve around the function (declaration slightly reformatted to include the class etc.):*
>
> */** Adds a point to the rubber band (in map coordinates) and to the capture list (in layer coordinates) */*
> *int QgsMapToolCapture::addVertex(const QgsPoint& point);*
>
> *It is called, for example, in QgsMapToolSplitFeatures::cadCanvasReleaseEvent(QgsMapMouseEvent * e), as following:*
>
> *addVertex(e->mapPoint());*
>
> *On its turn, QgsMapMouseEvent::snapPoint has several code paths, all with QgsSnappingUtils::snapToMap, to calculate the mMapPoint that is returned in mapPoint().*
>
> *Back to the function QgsMapToolCapture::addVertex. To create the capture list in layer coordinates, as promised in the comment, it uses*

> *QgsMapToolCapture::nextPoint, which contains the following line (including the comment):*
>
> *layerPoint = toLayerCoordinates(vlayer, mapPoint); //transform snapped point back to layer crs*
>
> *Thus, if I understand it correctly, the point is snapped (and transformed) to map coordinates and then transformed back to layer coordinates. I think this might be the problem, because floating point arithmetic doesn't guarantee that you end up with exactly the original value if you perform a calculation and then the inverse calculation.*

would you please raise this issue in the developers mailing list or do apull request in the qgis code repo on github? thanks!

#6 - 2015-11-19 12:17 AM - Daan Goedkoop
- Assignee set to Martin Dobias

Updated to reflect progress in pull request

#7 - 2015-11-24 03:40 AM - Bernd Eversmann

We have come across this issue as well, and the rounding errors cause other problems. For instance they affect the topology checker (which is a very crucial feature for us).
What happens is that because of rounding errors the topology checker shows a lot of gaps or overlaps which are not acually there, or which become only visible in extremely large scales because of a difference in the coordinates of the vertex in the 7th or 8th decimal.

I know this is a separate issue, but the tolerance setting in the topology checker doesn't seem to work, otherwise that could be a workaround.

Bernd

#8 - 2015-11-27 01:05 PM - Daan Goedkoop

Other examples of affected functions are topological editing and duplicate feature removal. I have made a screencast to demonstrate this (showing the expected behaviour first, and then the current behaviour): https://youtu.be/JB-Y88EwZC4

#9 - 2016-02-15 01:57 PM - Daan Goedkoop
- Status changed from Open to Closed

Files

Step1.png	237 KB	2015-11-03	Daan Goedkoop
Step2.png	37.6 KB	2015-11-03	Daan Goedkoop
Step3.png	255 KB	2015-11-03	Daan Goedkoop
Step4.png	47 KB	2015-11-03	Daan Goedkoop
Step5.png	372 KB	2015-11-03	Daan Goedkoop
Step6.png	152 KB	2015-11-03	Daan Goedkoop
bug.tgz	3.53 KB	2015-11-06	Daan Goedkoop