QGIS Application - Bug report #4088
Freeze while rendering labels (master 17e864b)

2011-07-15 12:36 AM - Sandro Santilli

Status:	Closed		
Priority:	Normal		
Assignee:	Martin Dobias		
Category:			
Affected QGIS version:		**Regression?:**	No
Operating System:		**Easy fix?:**	No
Pull Request or Patch supplied:	No	**Resolution:**	fixed
Crashes QGIS or corrupts data:		**Copied to github as #:**	14071

Description

While testing the fix for bug #4083 I got a qgis freeze.
Opened the city_data.qgs project file (created with https://github.com/strk/qgis_pgis_topoview for the city_data topology created by [postgis_src]/topology/test/load_topology.sql).
Zoomed in.

Last few lines on console are:

Debug: /usr/src/qgis/qgis/src/core/qgsmaprenderer.cpp: 371: (render) Input extent: 3.0000000000000000,-4.0000000000000000 : 63.0000000000000000,45.0000000000000000
Debug: /usr/src/qgis/qgis/src/core/qgsmaplayer.cpp: 924: (setCacheImage) cache Image set!
Debug: /usr/src/qgis/qgis/src/core/qgsvectorlayer.cpp: 957: (draw) rendering v2:
UNKNOWN RENDERER

Debug: /usr/src/qgis/qgis/src/core/qgsvectorlayer.cpp: 974: (draw) attrs: edge_id - 0
Debug: /usr/src/qgis/qgis/src/providers/postgres/qgspostgresprovider.cpp: 3433: (openCursor) Starting read-only transaction
Debug: /usr/src/qgis/qgis/src/core/qgspallabeling.cpp: 606: (registerFeature) Ignoring feature 30 due PAL exception: Geometry Type is unknow

Associated revisions

Revision 8e08c53a - 2011-07-20 12:45 AM - Martin Dobias

Fix #4088: better handling of unknown geometry types

History

#1 - 2011-07-15 12:37 AM - Sandro Santilli

Note: there's *no* CPU activity and memory usage is low. GUI is unresponsible, doesn't refresh (immune to expose X events) and mouse pointer is of the "busy" flavor.

#2 - 2011-07-15 12:52 AM - Sandro Santilli

Can't reproduce it :/

#3 - 2011-07-15 01:45 AM - Martin Dobias

- File bug4088fix.diff added

That sounds like a mutex not unlocked when PAL exception is thrown. I have prepared a small patch, though I am not at my dev machine now so I cannot compile/test it.

Most likely problem happens when a geometry collection is passed to labeling engine. The newly added clipping in #4083 might be the cause - the intersection operation between extent rectangle and input feature may result in geometry collection, right?

#4 - 2011-07-15 01:45 AM - Martin Dobias

- Assignee set to Martin Dobias

#5 - 2011-07-15 02:00 AM - Sandro Santilli

Intersections made with GEOS can result in collections.
They can also result in crashes so I'd be very careful about using them for
labeling purposes. If there's a way to avoid changing the input but only the way
it's considered it should be favored (ie: only consider segments intersecting the viewport rather than all of them?).

Btw, it would be helpful if the message showed also the geometry type id beside the 'unknow' label (there's a typo too)

#6 - 2011-07-19 03:44 PM - Martin Dobias

- Status changed from Open to Closed
- Resolution set to fixed
- Pull Request or Patch supplied set to No

Bugfix applied in commit:8e08c53

Regarding the intersections with GEOS: is there a high chance of a crash? Are there still robustness problems?

If we were going to use the whole segments that intersect the viewport we would still have to calculate the intersection of segments with the viewport in order to place label candidates only to visible positions. So I guess a better solution is to run intersect on the whole geometry.

#7 - 2011-07-20 12:35 AM - Sandro Santilli

I wouldn't say "high" chances, but there still are. Maybe you could catch and let go if it happens. I don't really have statistical data about robustness issues happening with rectangular intersections (should be handled in a special way, faster).

Doesn't the memory representation you're using (the Qt one) support clipping ?

#8 - 2011-07-20 02:25 AM - Martin Dobias

The memory representation for labeling is a custom one introduced by the PAL labeling library: geometries are input as GEOS geometry instances and then converted to that representation.

Anyway Qt is able to do clipping, but that is limited to rendering (and not exposed to public API) and seems to be slow.

Additionally QGIS has its own line and polygon clipping routines, but for yet another representation (using QPolygonF from Qt). Ultimately this may be the fastest and safest solution, however it would require us to make QgsClipper class more generic.

Files

bug4088fix.diff	1.24 KB	2011-07-14	Martin Dobias