Qt 3D Discussed

By *Roy Schestowitz*
Created *21/10/2019 - 4:27pm*
Submitted by Roy Schestowitz on Monday 21st of October 2019 04:27:55 PM Filed under Development [1] KDE [2]

- **Qt 3D Will Still Be Improved On Alongside Qt Quick 3D [3]**

  While Qt Quick 3D has been talked up a lot recently with The Qt Company's plans for that new 3D module inside the current Qt5 and future Qt6 tool-kits, Qt 3D itself is not going away.

  Qt Quick 3D will offer 3D support to Qt Quick via QML and C++ APIs but the existing Qt 3D support isn't going to be eliminated and in fact will be improved upon as we near the Qt 6.0 release in about one year's time.

- **The Future of Qt 3D [4]**

  As you will have read, a new module called Qt Quick 3D will begin offering 3D capabilities to Qt Quick via a QML API (and a planned C++ API for Qt 6). What does this mean for Qt 3D and where will it fit in the Qt ecosystem? Hopefully this blog post and the following one will help answer that question as well as give some insights into what we are working on in Qt 3D. This blog post will focus on the changes coming with Qt 5.x and the following article will details some of the research we are doing to improve Qt 3D on the Qt 6 timescale.

- **Qt 3D: One too many threads [5]**

  Qt 3D makes heavy use of threads, as a way to spread work across CPU cores and maximize throughput, but also to minimize the chances of blocking the main thread. Though nice on

paper, the last case eventually leads to added complexity. Sometimes, there are just one too many threads.

In the past, we?ve been guilty of trying to do too much within Qt 3D rather than assuming that some things are the developer?s duty. For instance there was a point in time where we?d compare the raw content of textures internally. The reason behind that was to handle cases where users would load the same textures several times rather than sharing one. This led to code that was hard to maintain and easy to break. Ultimately it provided convenience only for what can be seen as a misuse of Qt 3D, which was not the the original intention.

We had similar systems in place for Geometries, Shaders? Part of the reason why we made such choices at the time was that the border between what Qt 3D should or shouldn?t be doing was really blurry. Over time we?ve realized that Qt 3D is lower level than what you?d do with QtQuick. A layer on top of Qt 3D would have instead been the right place to do such things. We?ve solved some of these pain points by starting work on Kuesa which provides assets collections.

Development KDE

Links:
[1] http://www.tuxmachines.org/taxonomy/term/145
[2] http://www.tuxmachines.org/taxonomy/term/108
[3] https://www.phoronix.com/scan.php?page=news_item&px=Qt-3D-And-Qt-Quick-3D-Future
[4] https://www.qt.io/blog/the-future-of-qt-3d
[5] https://www.kdab.com/qt-3d-one-too-many-threads/