Qt World Summit 2016

October 18-20 | San Francisco, USA

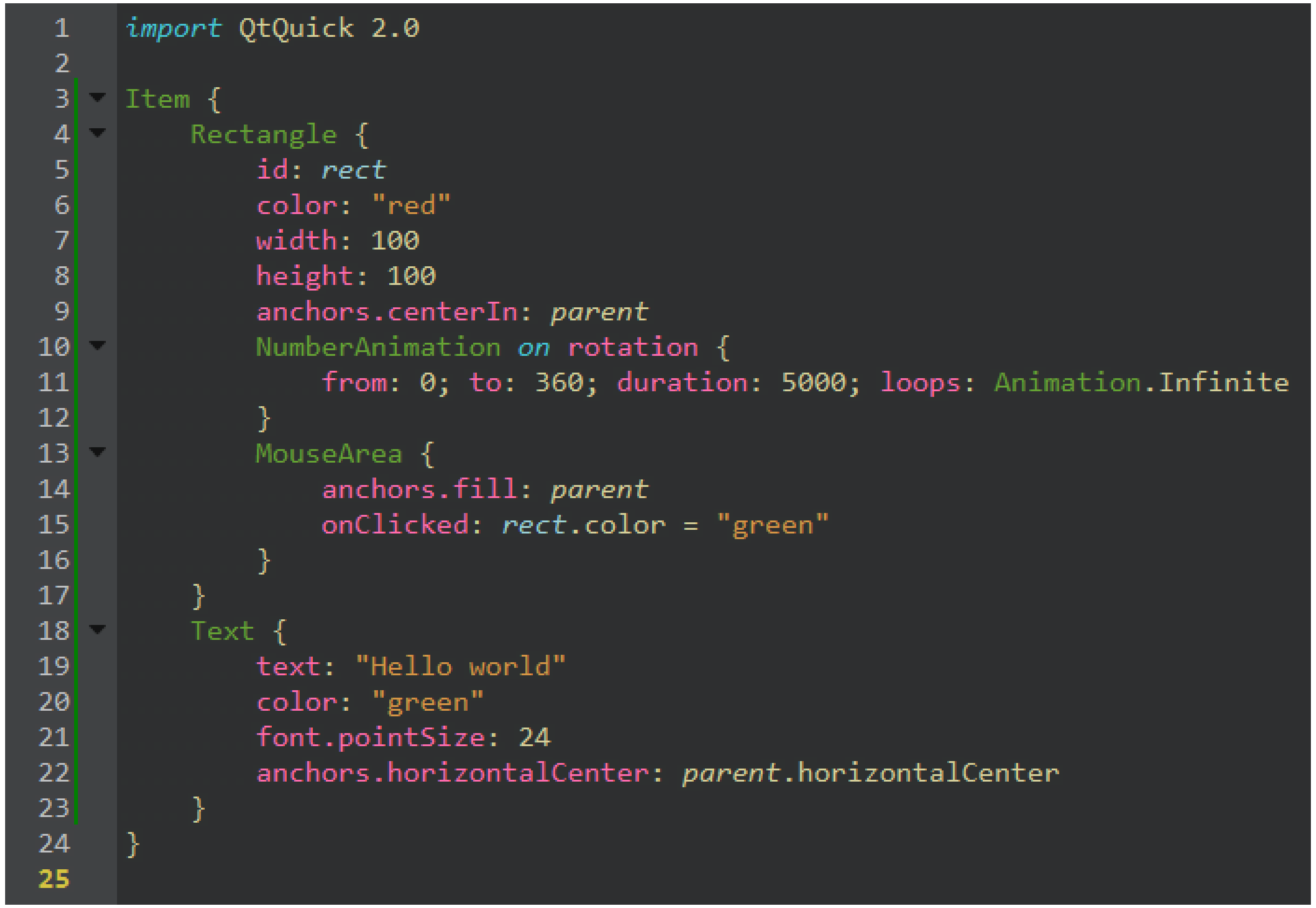
Qt Quick Scenegraph Advancements in Qt 5.8 and Beyond

László Agócs (lagocs / @alpqr)
Senior Software Engineer
The Qt Company, Oslo, Norway

Agenda

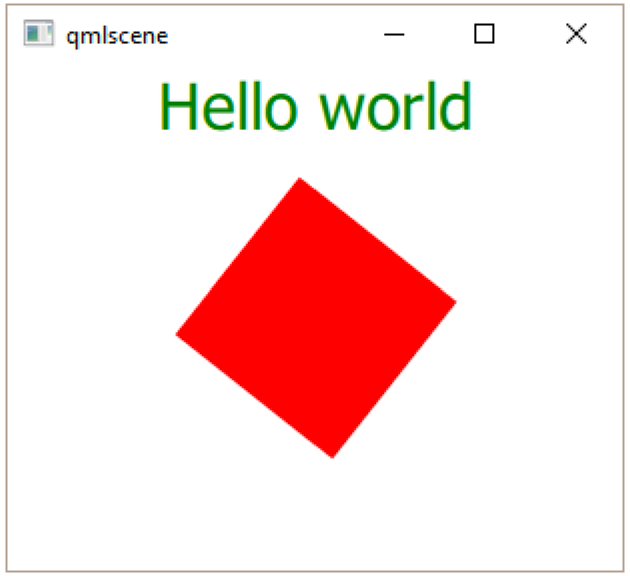
- Qt Quick modularization in Qt 5.8
- Integrated software backend
- Modern graphics APIs
- Consequences for applications
- Future plans

Qt Quick



```
import QtQuick 2.0

Item {
    Rectangle {
        id: rect
        color: "red"
        width: 100
        height: 100
        anchors.centerIn: parent
        NumberAnimation on rotation {
            from: 0; to: 360; duration: 5000; loops: Animation.Infinite
        }
        MouseArea {
            anchors.fill: parent
            onClicked: rect.color = "green"
        }
    }
    Text {
        text: "Hello world"
        color: "green"
        font.pointSize: 24
        anchors.horizontalCenter: parent.horizontalCenter
    }
}
```



Qt Quick and OpenGL

- Qt Quick 2 in Qt 5.0
    - Scene graph designed for OpenGL ES 2.0
    - Moved away from the QPainter world
    - Assumptions for OpenGL (2.0 + FBO or ES 2.0) being available everywhere

OpenGL on Desktop

- The latter was a bit too much to wish for
- Improvement attempt 1: ANGLE
  - D3D9, D3D11, or WARP on D3D11
- Improvement attempt 2: Mesa llvmpipe
- Improvement attempt 3: Dynamic selection at app startup
  - Driver blacklist

OpenGL on Embedded Low-end

Toradex Colibri iMX7 module

- No GPU
  - Or GPU with 2D-only acceleration
- Yes, this is still a thing!
- Software OpenGL not an option
- Qt Quick 2D Renderer
  - Initially targeted 2D (blit) accelerated systems
  - Idea: If only it did not need fullscreen updates on every frame…

New Graphics APIs

- Meanwhile: Metal, Direct3D 12, Vulkan
- Does not mean Qt and Qt Quick has to support all of them.
- But should have the possibility to do so, in case it becomes beneficial.
- Note: Qt Quick is not a 3D engine with millions of draw calls.
- Note:
```
basic enablers in Qt != support in Qt Quick
&& basic enablers in Qt != support in QPainter
```

New Graphics APIs

- Research and improvement opportunities:
  - Threading. Parallel command submission, texture management, etc.
  - Different approaches to shaders. More pre-compilation.
    - D3D shader bytecode, SPIR-V
  - Better tooling and system integration due to being the platform vendor’s primary API.
  - Lower driver CPU overhead.

Qt Quick in Qt 5.8

- No strict OpenGL requirement for the scenegraph anymore.

- Qt Quick no longer skipped when OpenGL is disabled in the Qt configuration
  - -no-opengl
  - missing GL headers/libs

- Builds on the existing concept of scenegraph plugins.
  - Software backend built-in
  - Experimental D3D12 backend as a plugin

Qt 5.8 – Software Backend

- All candidate features implemented:
  - Built-in, LGPLv3 + Commercial license, like the core of Qt.
  - No more hackish setup steps.
  - **Partial updates**!
  - Enable with:
    - Environment variable: `QT_QUICK_BACKEND=software`
    - In main(): `QQuickWindow::setSceneGraphBackend(QSGRendererInterface::Software)`
    - Implicit in no-opengl builds

Qt 5.8 – Software Backend

- Not the same as a software OpenGL rasterizer.
    - Much more lightweight.
    - Goes through the scene and makes ordinary QPainter calls.
- Suitable for the embedded low-end.
- And with 5.8 for desktop apps as well.
- Less complete, obviously, no shader effects and particles for instance.
    - But still great for many typical user interfaces.

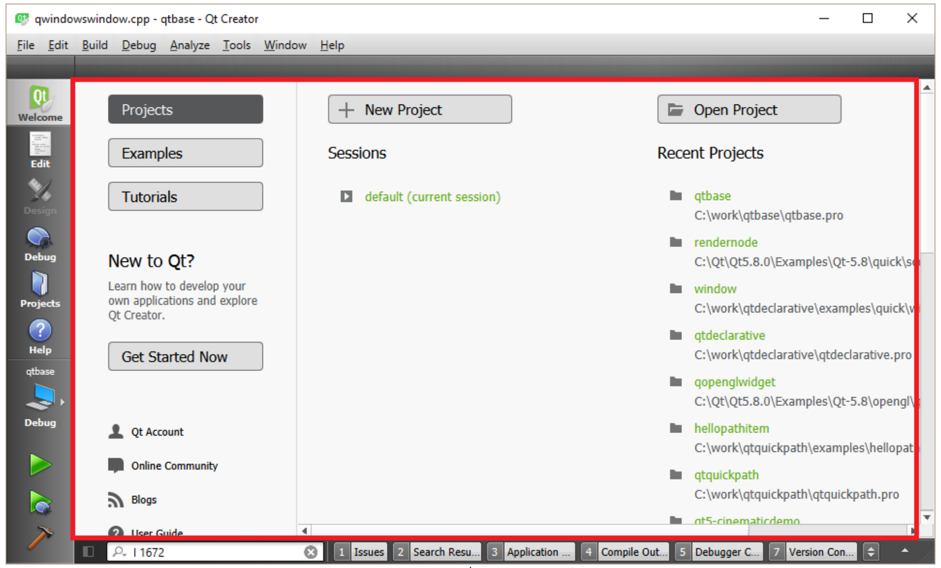
Qt 5.8 – Direct3D 12 Backend

- Experimental, proof of concept. But is rather stable.
- Plugin comes with Qt Quick
  - Built automatically on Windows 10 with MSVC 2015
  - Comes with the MSVC packages
- Not just traditional Win32, but UWP (WinRT) too.
- OpenGL is still the default (except in no-opengl builds)
  - `QT_QUICK_BACKEND=d3d12`
  - `QQuickWindow::setSceneGraphBackend(QSGRendererInterface::Direct3D12)`

Qt 5.8 – Direct3D 12 Backend

- Performance
- HLSL
  - Built-in materials: HLSL 5.0, pre-compiled to D3D shader bytecode at build time via fxc
  - ShaderEffect: source or bytecode from files
    - File selectors! See ShaderEffect docs.
- Mipmaps, multisampling, semi-transparent windows
- Robustness

Qt 5.8 – non-GL Backend Limitations

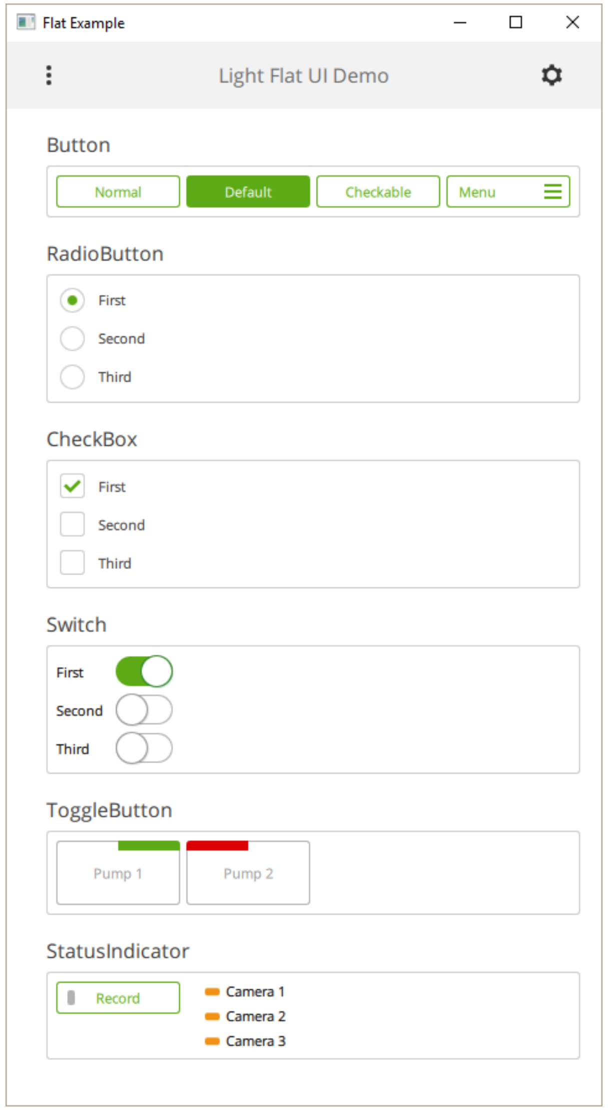

qwindowswindow.cpp - qtbase - Qt Creator
File Edit Build Debug Analyze Tools Window Help
Welcome
Edit
Design
Debug
Projects
Help
qtbase
Debug
Projects
Examples
Tutorials
New to Qt?
Learn how to develop your own applications and explore Qt Creator.
Get Started Now
Qt Account
Online Community
Blogs
User Guide
New Project
Sessions
default (current session)
Open Project
Recent Projects
qtbase
C:\work\qtbase\qtbase.pro
rendernode
C:\Qt\Qt5.8.0\Examples\Qt-5.8\quick\s
window
C:\work\qtdeclarative\examples\quick\w
qtdeclarative
C:\work\qtdeclarative\qtdeclarative.pro
qopenglwidget
C:\Qt\Qt5.8.0\Examples\Qt-5.8\opengl\
hellopathitem
C:\work\qtquickpath\examples\hellopat
qtquickpath
C:\work\qtquickpath\qtquickpath.pro
qt5-cinematicdemo
l 1672
1 Issues
2 Search Resu...
3 Application ...
4 Compile Out...
5 Debugger C...
7 Version Con...

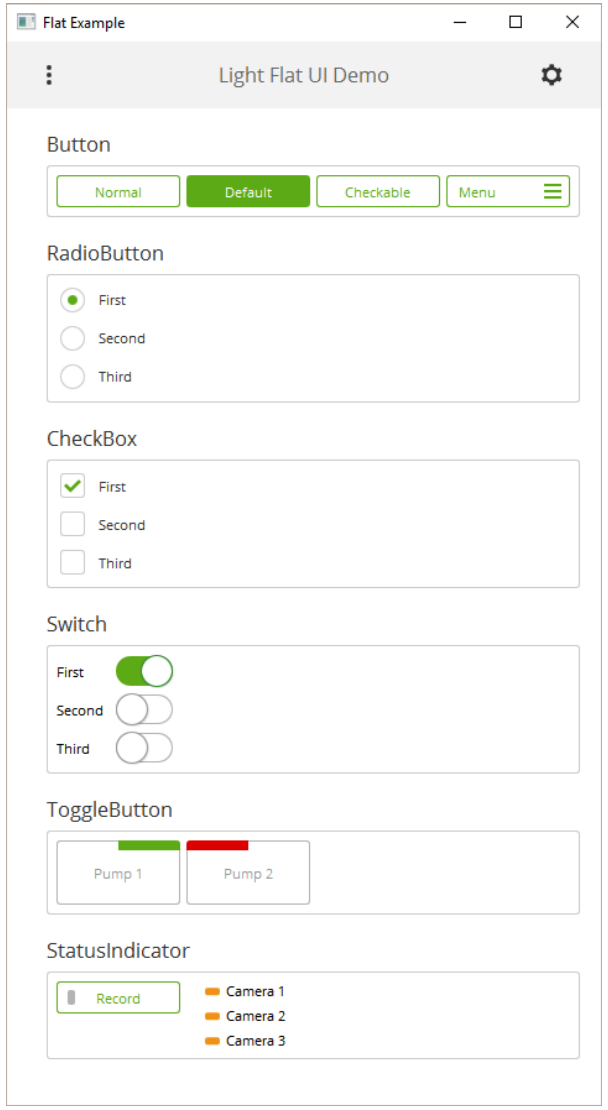
- What's missing?
  - Particles
  - Distance field based text rendering
  - QQuickFramebufferObject / Canvas via FBO
  - Texture atlases
  - Point/lines with width != 1
  - Rendering via QSGEngine/QSGAbstractRenderer

  - No QQuickWidget with D3D12, OK with Software
  - No shaders with Software, OK with D3D12

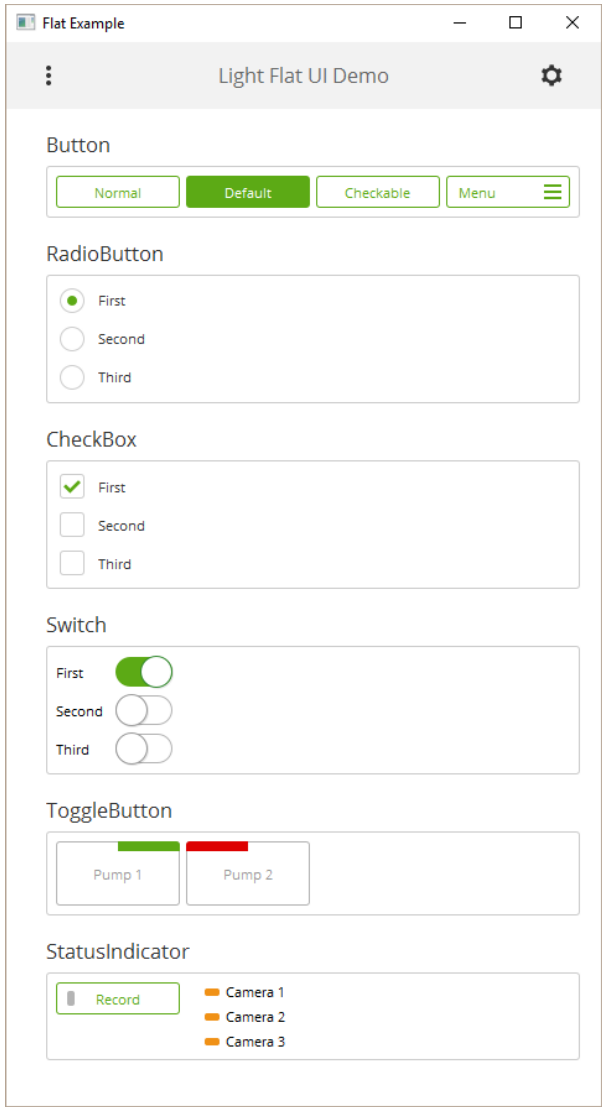
SW QQuickWidget is new in 5.8: **mix QWidget and QML without involving OpenGL dependencies**

- Will fail gracefully in most cases, e.g. particles are just not shown.

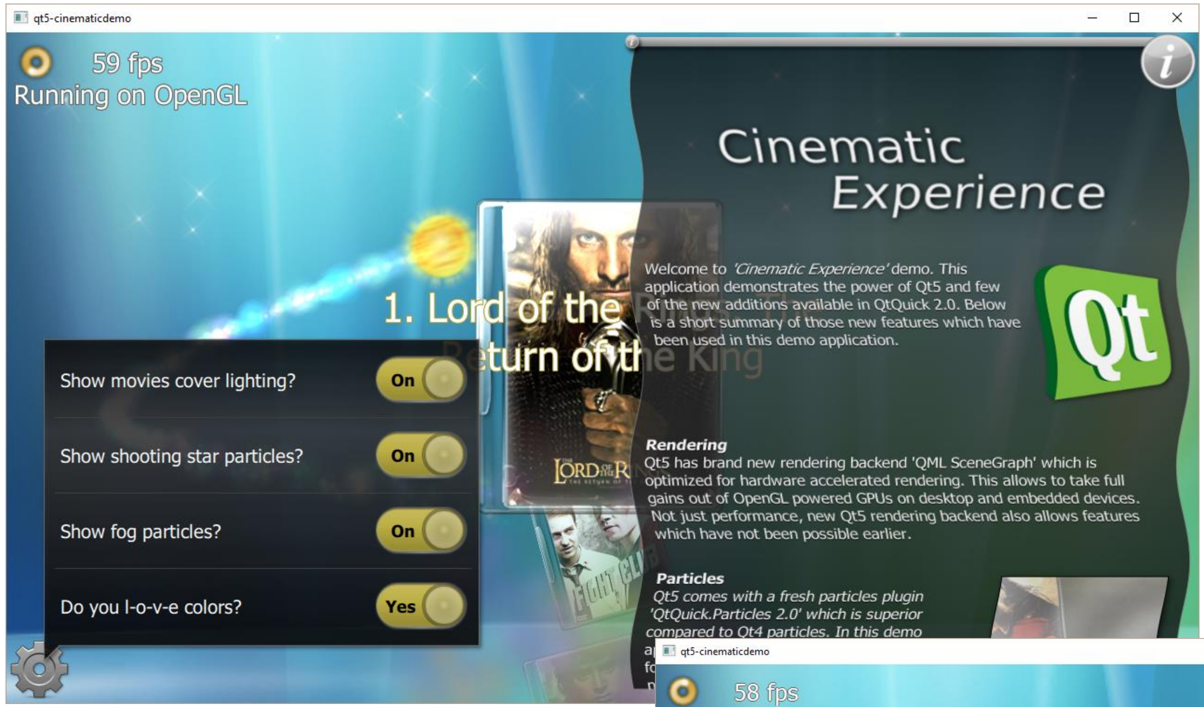




```
set QT_QUICK_BACKEND=d3d12
```



```
set QT_QUICK_BACKEND=software
```



qt5-cinematicdemo

59 fps
Running on OpenGL

Cinematic
Experience

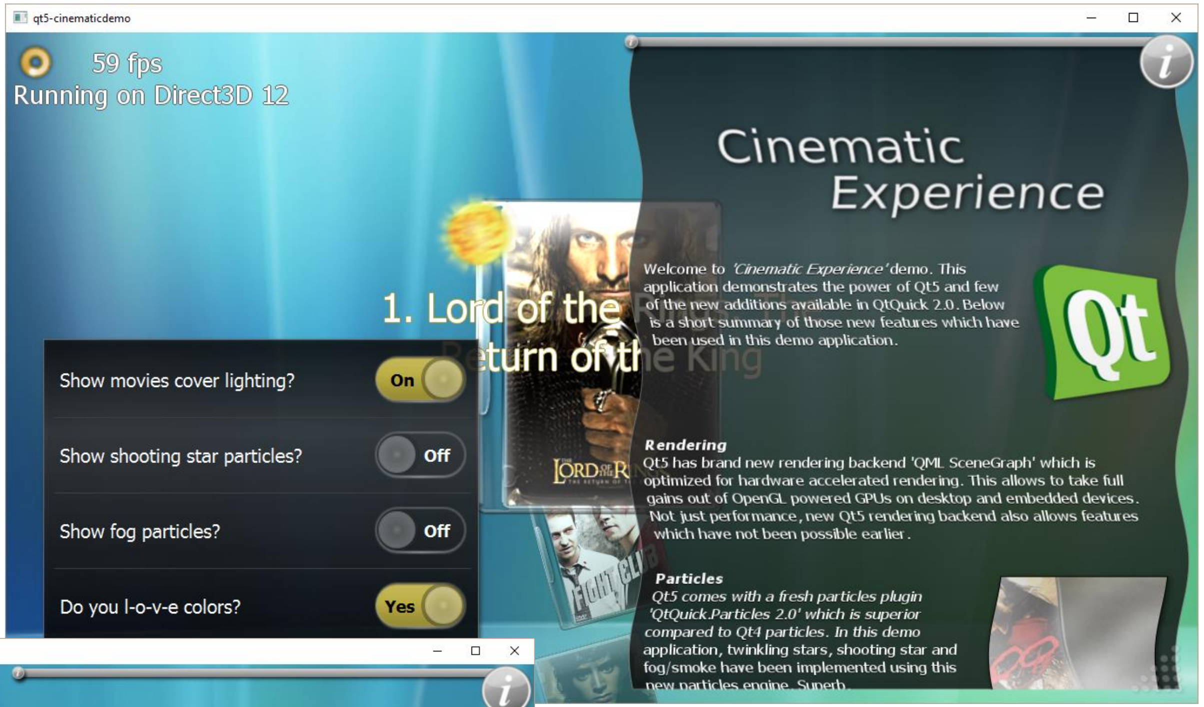
1. Lord of the Rings: The Return of the King

Welcome to *'Cinematic Experience'* demo. This application demonstrates the power of Qt5 and few of the new additions available in QtQuick 2.0. Below is a short summary of those new features which have been used in this demo application.

Rendering
Qt5 has brand new rendering backend 'QML SceneGraph' which is optimized for hardware accelerated rendering. This allows to take full gains out of OpenGL powered GPUs on desktop and embedded devices. Not just performance, new Qt5 rendering backend also allows features which have not been possible earlier.

Particles
Qt5 comes with a fresh particles plugin 'QtQuick.Particles 2.0' which is superior compared to Qt4 particles. In this demo

Show movies cover lighting? On
Show shooting star particles? On
Show fog particles? On
Do you l-o-v-e colors? Yes



qt5-cinematicdemo

59 fps
Running on Direct3D 12

Cinematic
Experience

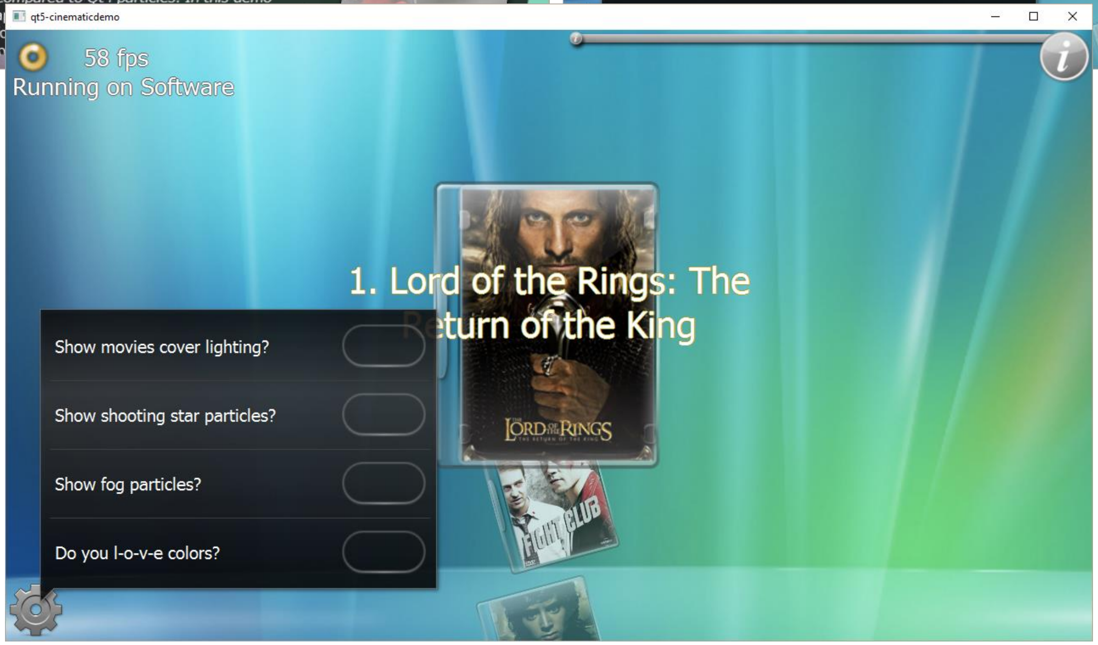
1. Lord of the Rings: The Return of the King

Welcome to *'Cinematic Experience'* demo. This application demonstrates the power of Qt5 and few of the new additions available in QtQuick 2.0. Below is a short summary of those new features which have been used in this demo application.

Rendering
Qt5 has brand new rendering backend 'QML SceneGraph' which is optimized for hardware accelerated rendering. This allows to take full gains out of OpenGL powered GPUs on desktop and embedded devices. Not just performance, new Qt5 rendering backend also allows features which have not been possible earlier.

Particles
Qt5 comes with a fresh particles plugin 'QtQuick.Particles 2.0' which is superior compared to Qt4 particles. In this demo application, twinkling stars, shooting star and fog/smoke have been implemented using this new particles engine. Superb.

Show movies cover lighting? On
Show shooting star particles? Off
Show fog particles? Off
Do you l-o-v-e colors? Yes



qt5-cinematicdemo

58 fps
Running on Software

1. Lord of the Rings: The Return of the King

Show movies cover lighting?
Show shooting star particles?
Show fog particles?
Do you l-o-v-e colors?

Other Modules?

- Some Qt modules outside QtGui and QtQuick are tied to OpenGL.
- Do not expect Qt3D, QtCanvas3D, QtGraphicalEffects, etc. to work through D3D12 or Software.
- Notable exceptions:
  - QtWebEngine is fine with Software
  - QtLocation (maps) partially functional both with Software and D3D12

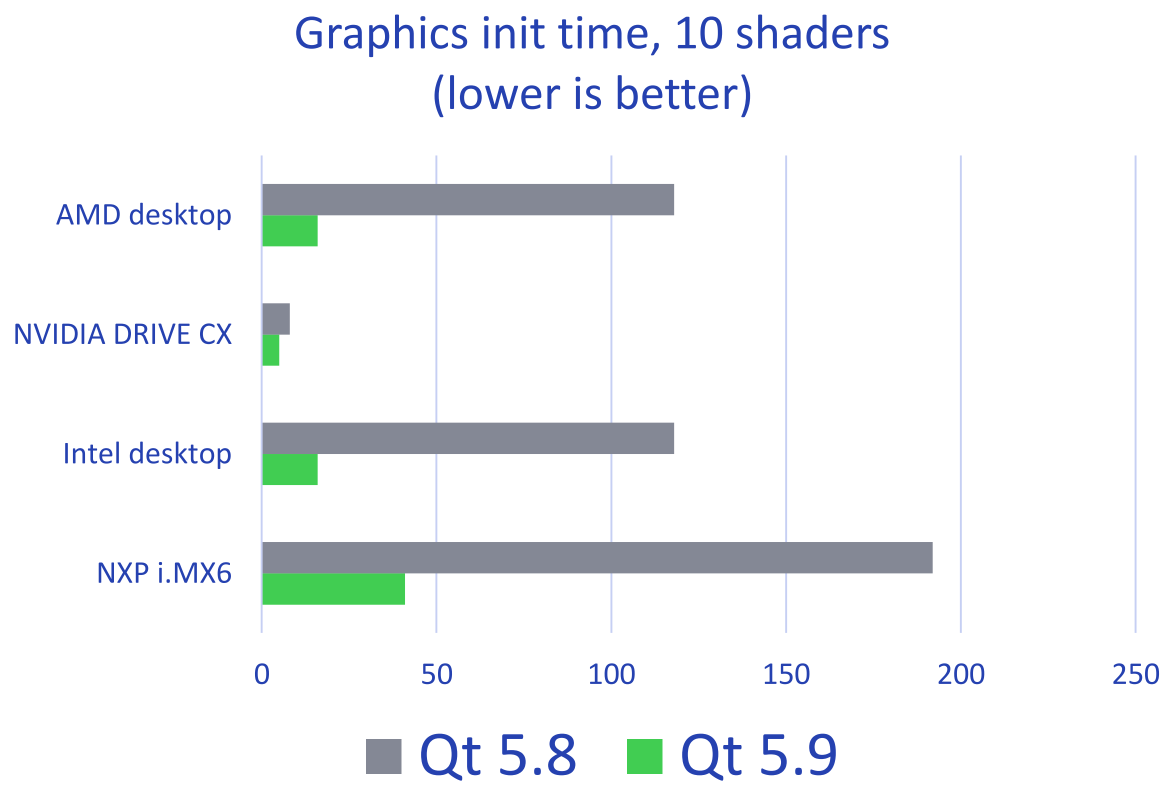
The Trouble with Cross-API Custom Items

- So you are doing custom (ItemHasContents) QQuickItems.
- There's a catch: materials are backend-specific (or may not even exist)
  - Utility APIs tied to GL: QSGFlatColorMaterial, QSGVertexColorMaterial, …
  - Same for QSGMaterialShader and most QSGSimple* helper classes

- Options for now (all new in 5.8):
  - Use QSGRectangleNode and QSGImageNode via QQuickWindow to get simple rectangle and image nodes that work everywhere.
  - Use QSGRenderNode to do backend-specific (GL/D3D12/QPainter/…) custom rendering.

Future Plans

- Vulkan? Not yet.
  - Basic enablers (QVulkanWindow) maybe
- OpenGL improvements?
  - Yes!
  - Shader program binary caching
  - Compressed textures?
  - More modern GLES 3.1/3.2 features in the scenegraph?

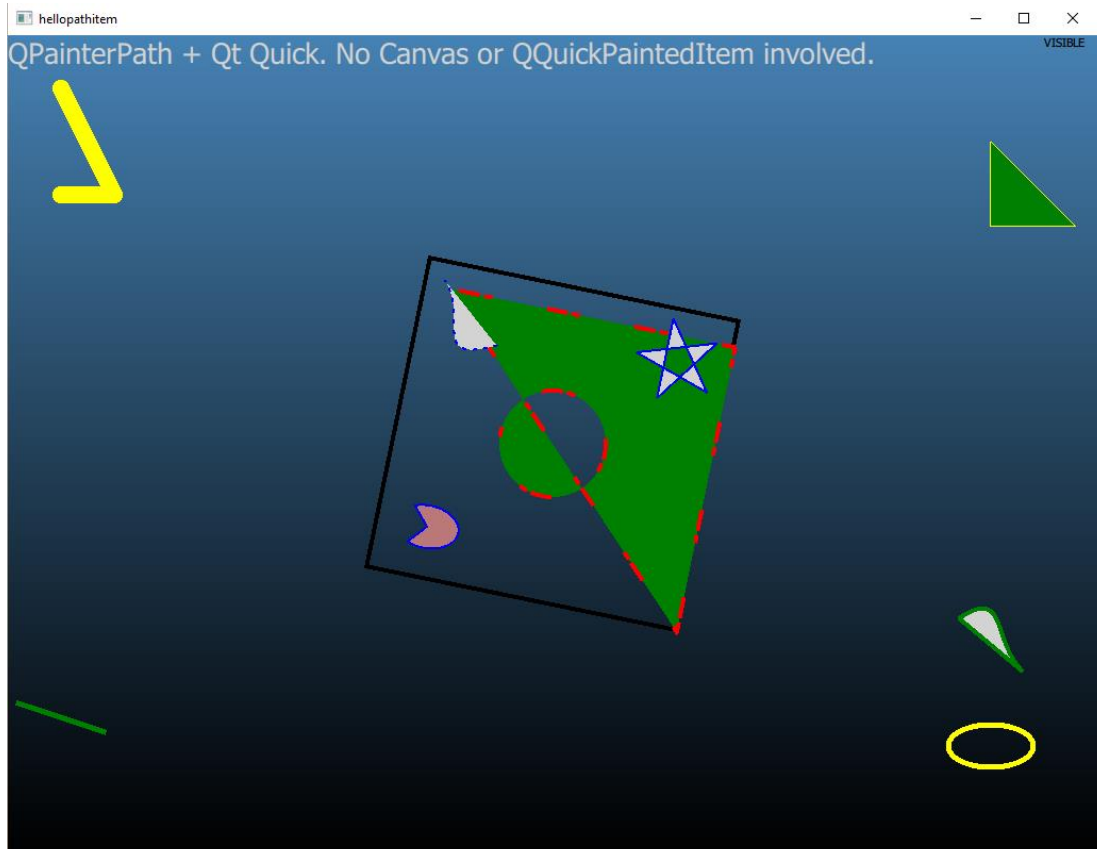


Future Plans

- OpenVG
  - No, not a joke
- Non-accelerated Linux framebuffer support needs to be renewed
  - DRM dumb buffers in addition to fbdev
- Printing of Qt Quick scenes?
  - Software backend can output to arbitrary QPaintDevice
  - Missing some plumbing to support printing

Future Plans

- Path rendering (shapes) ?
- Think adding a QPainterPath to a QML scene
  - Without a Canvas or QQuickPaintedItem
  - No expensive blits
  - Declarative API
- Multi-backend
  - Generic (reuse triangulator from QPainter)
  - Something more GPU/shader oriented?
  - **GL_NV_path_rendering**
  - QPainter for SW backend, OpenVG, etc.



QPainterPath + Qt Quick. No Canvas or QQuickPaintedItem involved.

Thank You!

https://qt.io
https://doc-snapshots.qt.io/qt5-5.8/qtquick-index.html
https://doc-snapshots.qt.io/qt5-5.8/qtquick-visualcanvas-scenegraph.html
https://doc-snapshots.qt.io/qt5-5.8/qtquick-visualcanvas-adaptations.html