What's New (and better) in Qt

Alistair Adams

24th May 2017

Qt Wayland

- › Multiprocess for embedded
- › Fully supported in Qt 5.8

Qt Wayland Compositor

Enables your device to be a complete SW platform

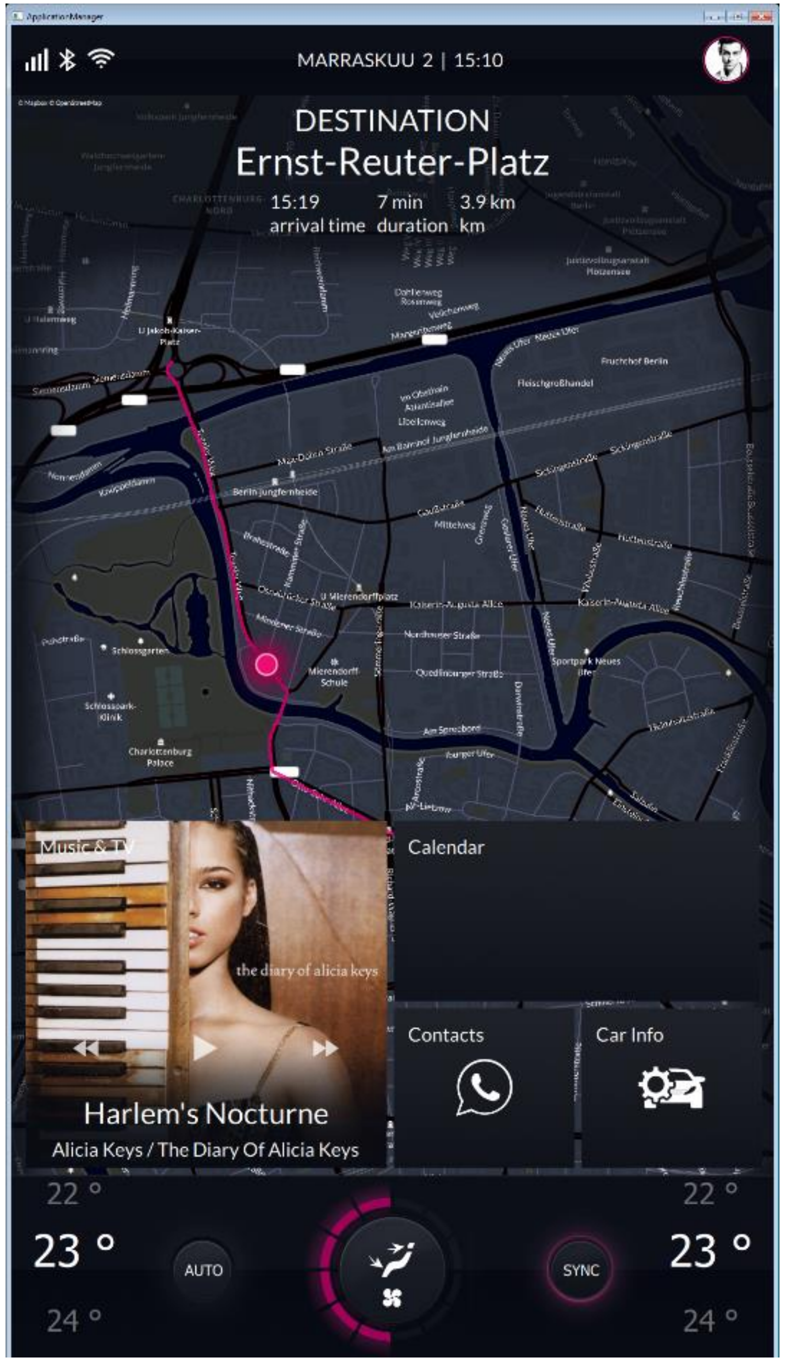
Create your unique platform

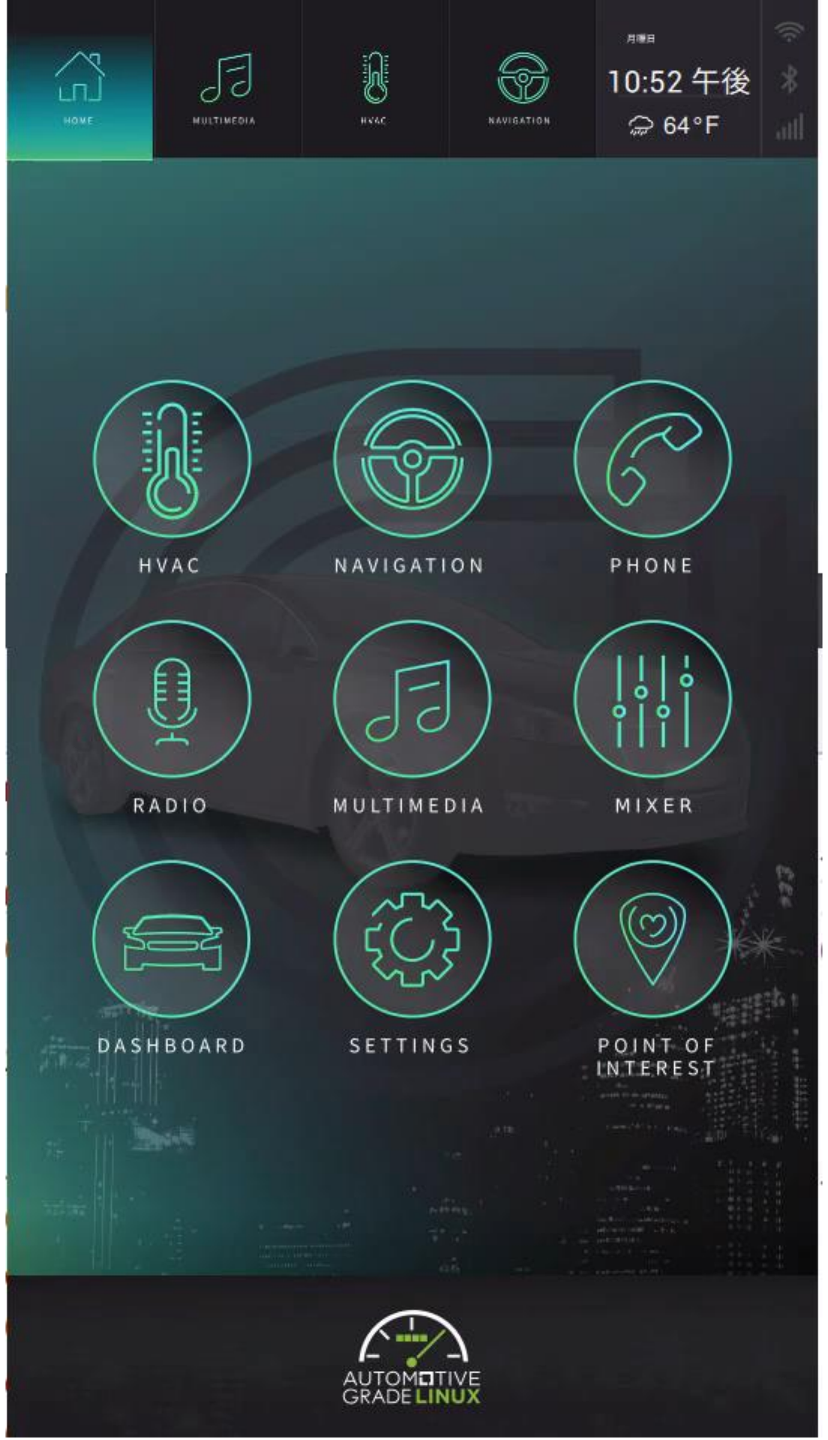
- Create multi-process embedded devices with Qt
- Improve robustness by splitting your system into different functional units/applications allowing you to make your device a SW platform for additional content

The developer experience with productivity at its core

- Less lines of code to test, debug and maintain - creating window servers on high abstraction levels
- Enable development & testing simultaneously by separating your UI into different functional units
- Minimize risk by only updating one unit at a time

Extended with Application Manager in Qt Automotive Suite





AGL Home Screen With Qt Compositor

```
    Transition {
        id: inTransition
        NumberAnimation {
            properties: "opacity, scale"
            from: 0
             to:1
             duration: 250
          }
    }
     Transition {
         id: outTransition
         NumberAnimation {
             properties: "opacity"
             from: 1
             to:0
             duration: 250
          }
         NumberAnimation {
             properties: "scale"
             from: 1
             to:5
             duration: 250
          }
     }
```

Qt Lite

- New configuration system for Qt
- Fully supported in Qt 5.8

Qt Lite

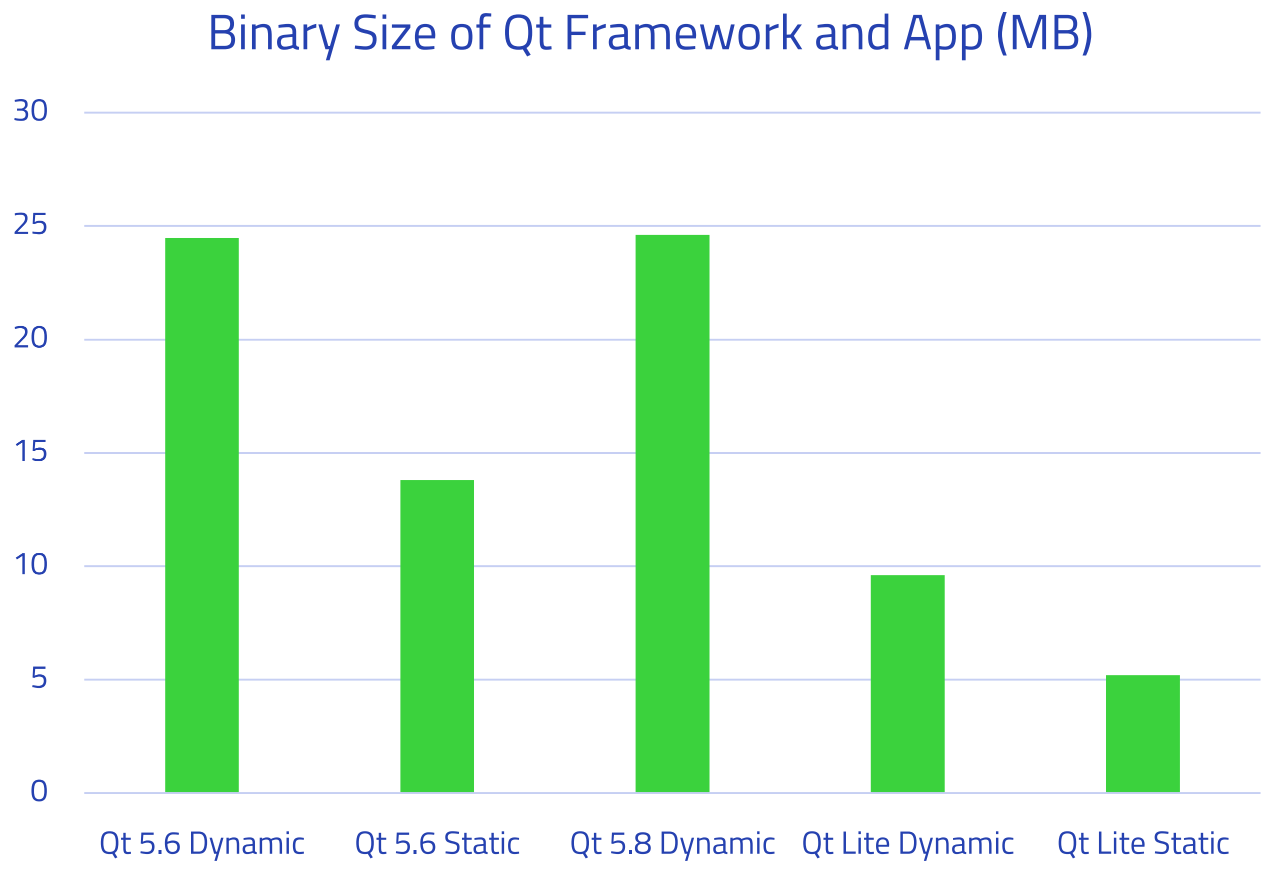
- Many low end devices where Qt is a good solution
  - …but minimum hardware requirements for Qt a bit high
    - Might not support OpenGL
    - Memory constrained

- Solution
  - Rearchitect scenegraph to allow different backends
    - Software rasterizer
    - OpenVG (Qt 5.9)
  - Reduce Qt footprint by making Qt more configurable
  - GUI tool to easily select only the required components
  - Documentation: Guidelines, recommendations, HW requirements

Qt Lite – Significant Improvements

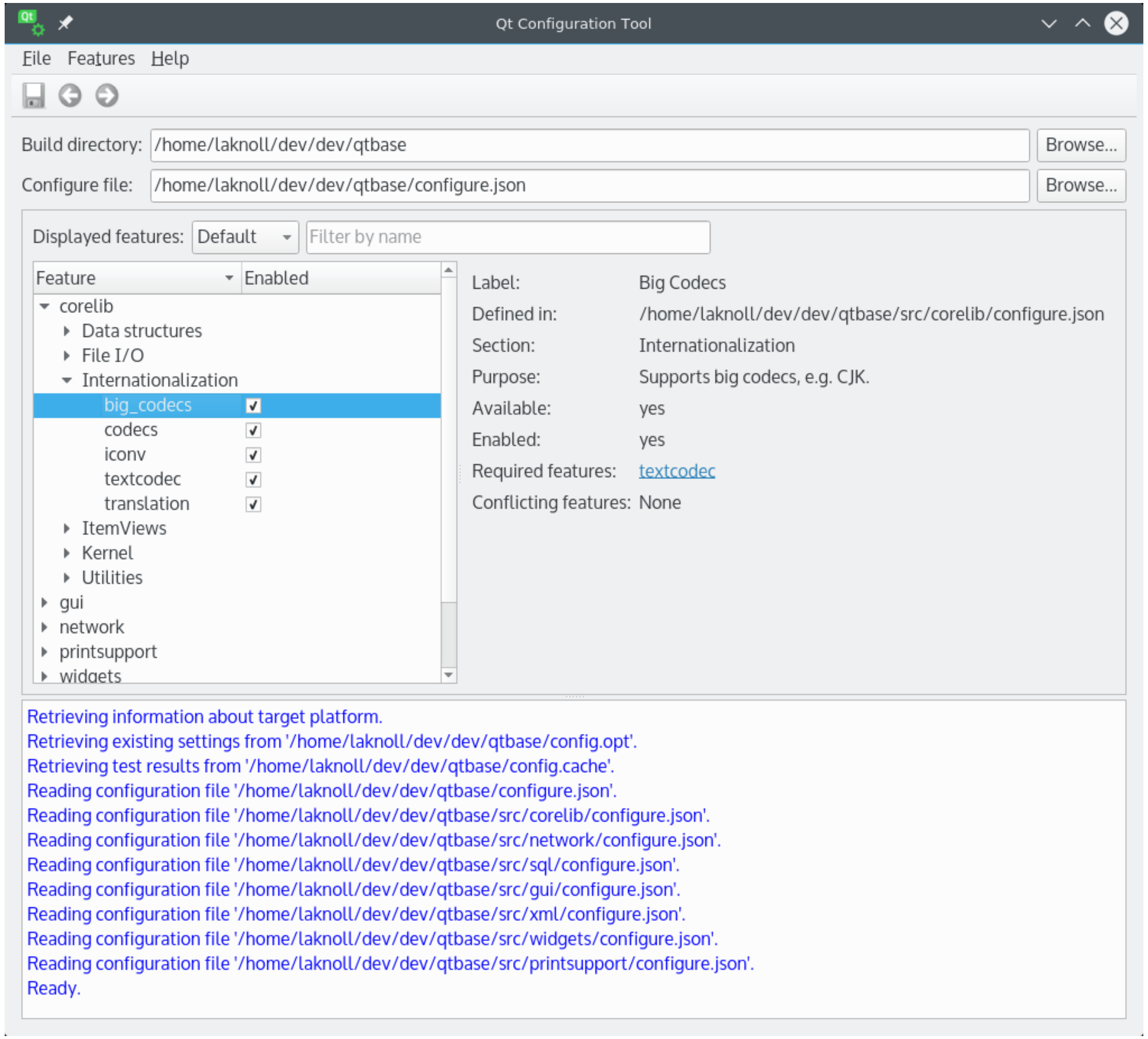
- Up to 80 percent smaller than Qt 5.6
- 65 percent smaller than Qt 5.6 static builds

- Faster application loading time
  - Static linking reduces application loading time.
  - It always guarantees constant loading time.
- Extreme flexibility to customize HMI & Qt library
  - Best optimized memory foot-print
  - It can make Qt libraries compact on a function level.
- Performance
  - Compiles only needed part of a library per application level.
  - Single process architecture will get the most performance benefit.
- Compatibility
  - It eases concerns of compatibility issue by pre-installed libraries.



Qt Lite Configuration Tool

- Easily see all available features
- Understand what you are doing with our integrated documentation
- Make it an integrated and simple part of your workflow
- Configuring Qt is no longer just for those with deep internal Qt knowledge



Build directory: /home/laknoll/dev/dev/qtbase

Configure file: /home/laknoll/dev/dev/qtbase/configure.json

Displayed features: Default

Filter by name

Feature	Enabled
corelib	
Data structures	
File I/O	
Internationalization	
big_codecs	✓
codecs	✓
iconv	✓
textcodec	✓
translation	✓
ItemViews	
Kernel	
Utilities	
gui	
network	
printsupport	
widgets	

Label:	Big Codecs
Defined in:	/home/laknoll/dev/dev/qtbase/src/corelib/configure.json
Section:	Internationalization
Purpose:	Supports big codecs, e.g. CJK.
Available:	yes
Enabled:	yes
Required features:	textcodec
Conflicting features:	None

```
Retrieving information about target platform.
Retrieving existing settings from '/home/laknoll/dev/dev/qtbase/config.opt'.
Retrieving test results from '/home/laknoll/dev/dev/qtbase/config.cache'.
Reading configuration file '/home/laknoll/dev/dev/qtbase/configure.json'.
Reading configuration file '/home/laknoll/dev/dev/qtbase/src/corelib/configure.json'.
Reading configuration file '/home/laknoll/dev/dev/qtbase/src/network/configure.json'.
Reading configuration file '/home/laknoll/dev/dev/qtbase/src/sql/configure.json'.
Reading configuration file '/home/laknoll/dev/dev/qtbase/src/gui/configure.json'.
Reading configuration file '/home/laknoll/dev/dev/qtbase/src/xml/configure.json'.
Reading configuration file '/home/laknoll/dev/dev/qtbase/src/widgets/configure.json'.
Reading configuration file '/home/laknoll/dev/dev/qtbase/src/printsupport/configure.json'.
Ready.
```

Qt Quick Compiler

- Next generation Qt Quick Compiler with Qt 5.8 and 5.9
- Previous one continues to be supported for commercial users

Qt Quick Compiler

In the beginning
QML parsed and
compiled at run-time,
V8 Javascript engine.

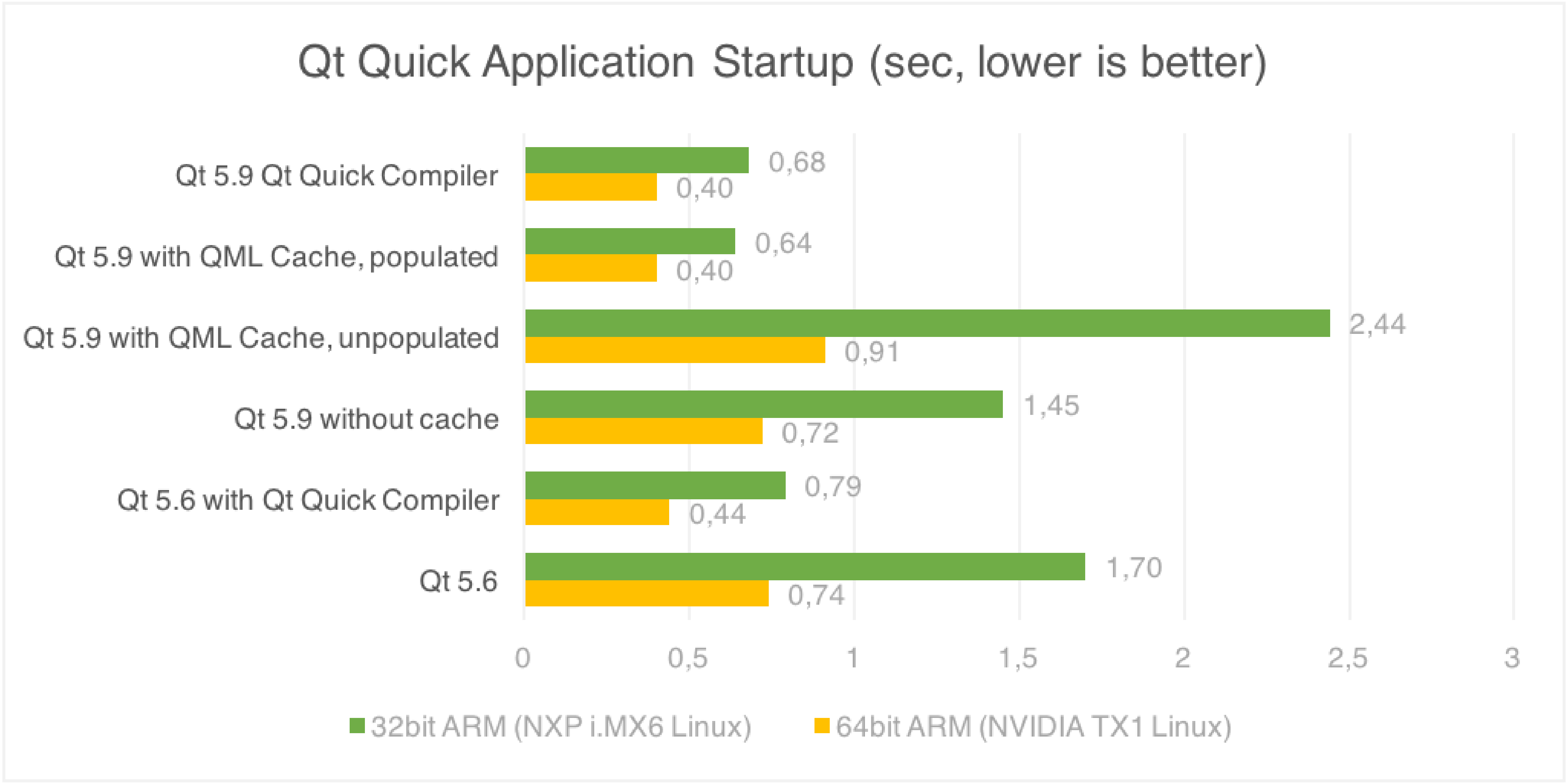
Qt 5.2
V4 Javascript engine.
Faster for the small JS
fragments in QML

Qt 5.3.1
Build time compiler
(qmlcompiler –
Commercial license only)
Much faster startup time

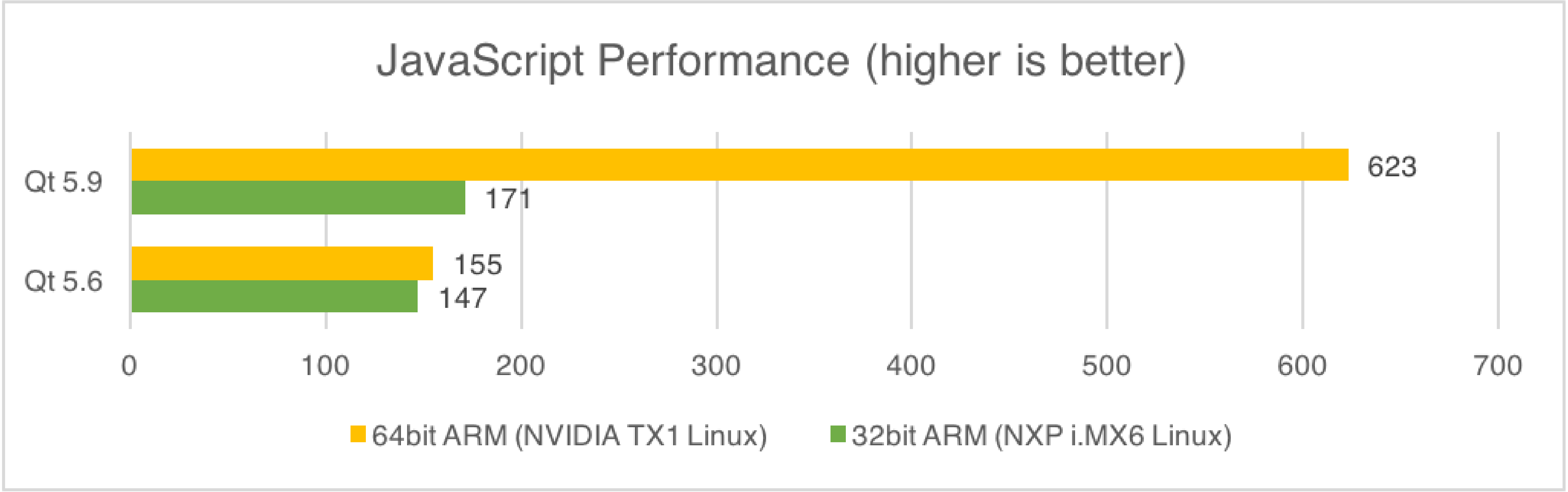
Qt 5.8
JIT Disc Cache .qmlc, .jsc files
2nd + runs have faster startup; up
to 60-70%, 30-40% typical
Also lower memory consumption.

Qt 5.9
Build-time JIT Cache creation (TP)
Fast first time startup,
IPR protection – QML hidden

Summary: 5.9 Options:
1. Run-time disc cache makes 2nd and subsequent startups faster
2. Qt 5.9 technical preview enables build-time creation of cache files
3. qmlcompiler available for commercial licensees.


Qt Quick Application Startup (sec, lower is better)
Qt 5.9 Qt Quick Compiler
0,68
0,40
Qt 5.9 with QML Cache, populated
0,64
0,40
Qt 5.9 with QML Cache, unpopulated
2,44
0,91
Qt 5.9 without cache
1,45
0,72
Qt 5.6 with Qt Quick Compiler
0,79
0,44
Qt 5.6
1,70
0,74
0
0,5
1
1,5
2
2,5
3
🟩 32bit ARM (NXP i.MX6 Linux)
🟨 64bit ARM (NVIDIA TX1 Linux)

JavaScript Performance



- 32-bit ARM is 16% faster
- 64-bit ARM 302% (i.e. 4x improvement).
  - Why? Qt 5.9 fully supports 64-bit ARM processors

QML Garbage Collector

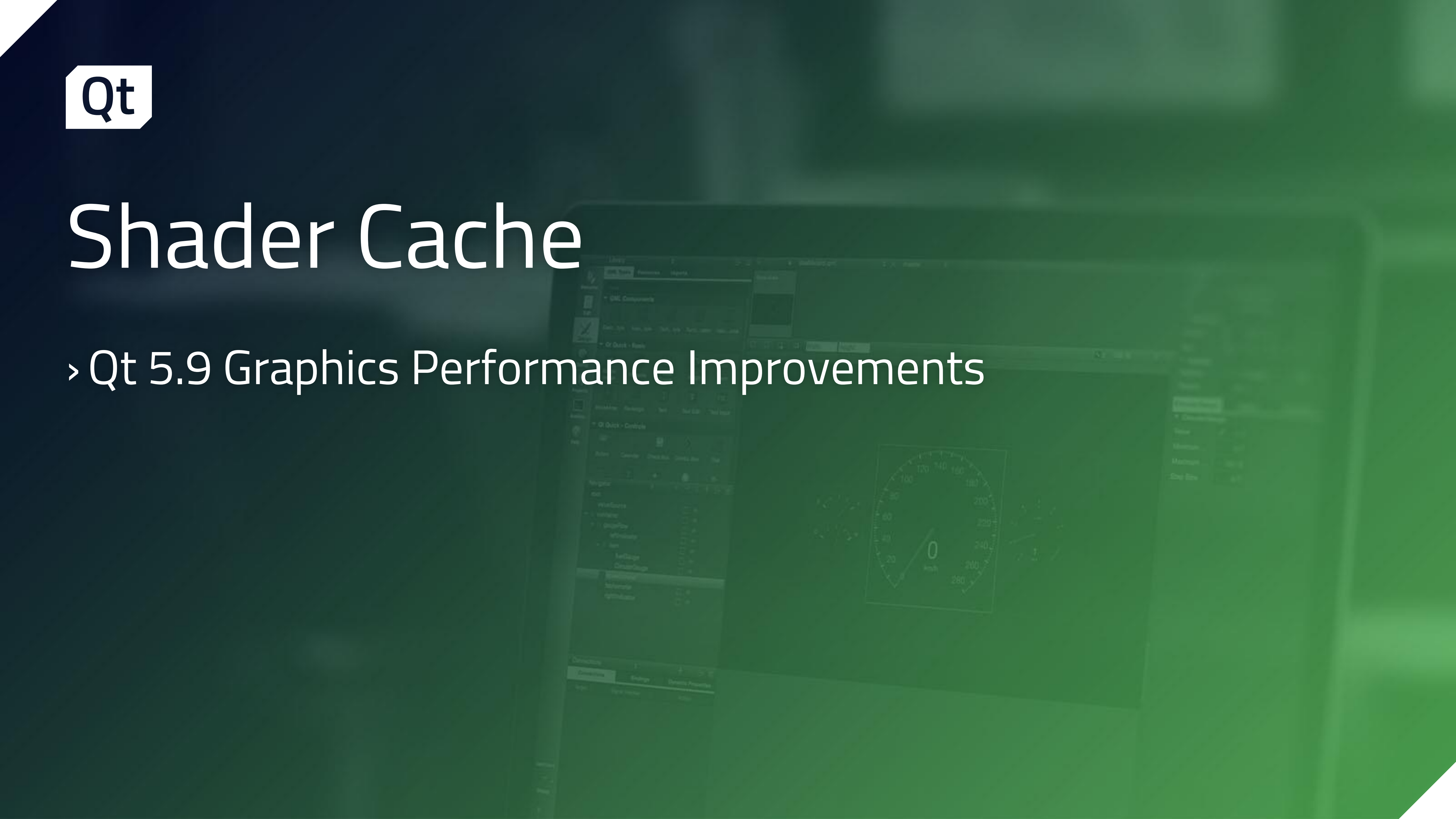
- Rewritten to provide better memory consumption and improved, more predictable performance for JavaScript code

Complete re-write of QML Garbage Collector

- Different data structures for allocating memory

- Amongst it advantages are:
  - Less overhead, faster allocation
  - Faster sweeping of the GC heap
  - More likely to return memory to the OS than the GC in 5.6
  - Less memory fragmentation inside the GC

- By the Numbers
  - 10-30% memory savings on the GC heap
  - 10-15% better performance

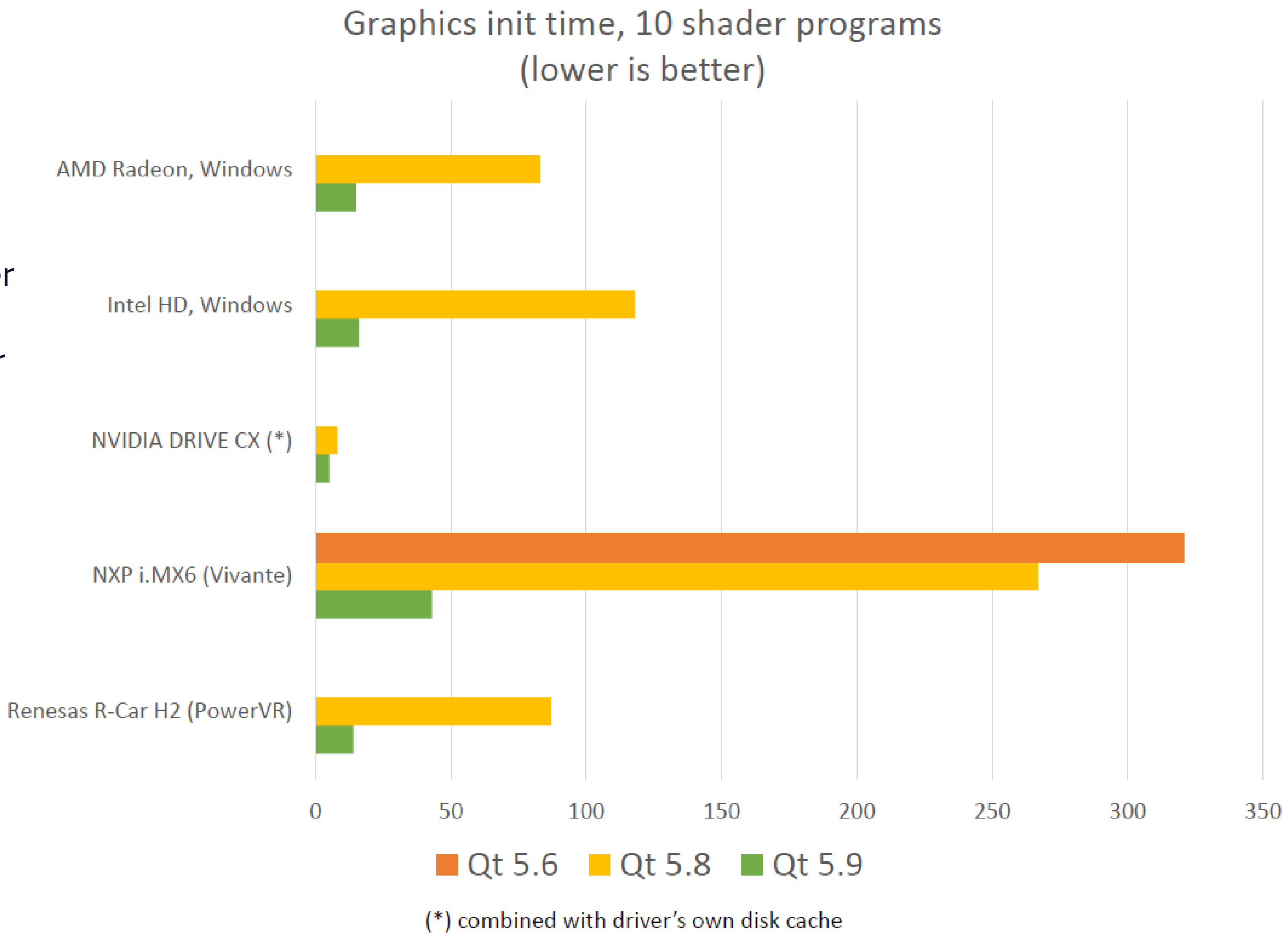


Shader Cache

› Qt 5.9 Graphics Performance Improvements

Shader Cache

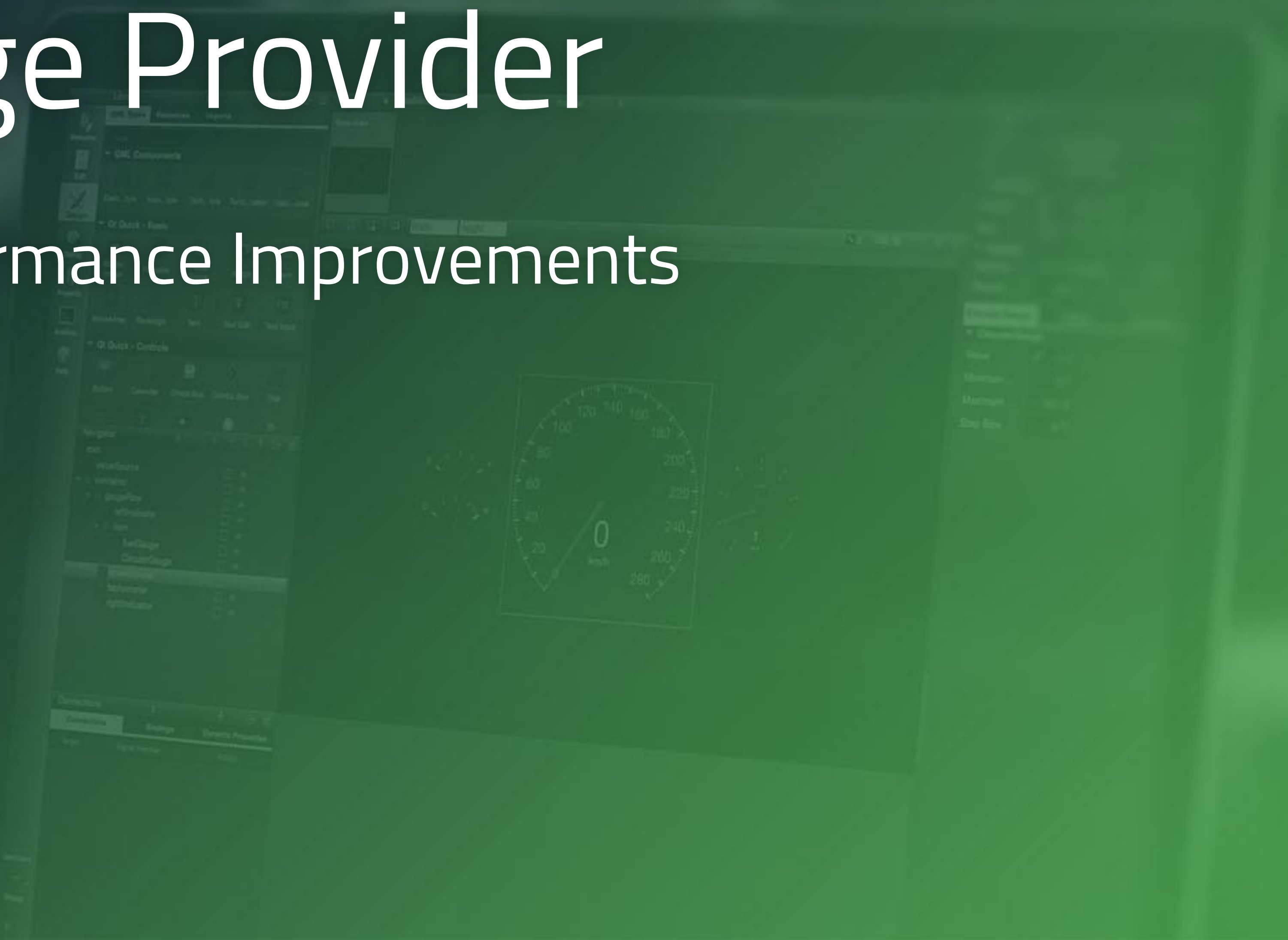
- New Shader cache feature
  - Qt specific binary cache for compiled shaders
  - Major startup time improvements after running the application once, even on drivers that actually implement shader caches themselves.



(*) combined with driver's own disk cache

Shared Image Provider

› Qt 5.9 Graphics Performance Improvements



Shared Image Provider: Saves disk and memory space in multi-process systems

- Built into QQuickImageProvider.
- QML example:

```
Image {
     source: "image://shared/usr/share/wallpapers/mybackground.jpg‶
}
```

- Looks for `/usr/share/wallpapers/mybackground.jpg`
- First process requesting file reads the image using normal Qt image loading.
- Decoded image data placed in shared memory, using the full file path as key.
- Later processes requesting the same image will not need to load again
- Shared image data kept until the last process has deleted
- If system memory sharing is not available, falls back to normal, unshared image loading.

Share Resources in Graphics Memory

- Shared Image Provider helps with multi-process UIs
- Bigger problem is shared resources in graphics memory
- Graphics driver dependent
- On-going research for Qt 5.11

Qt SCXML

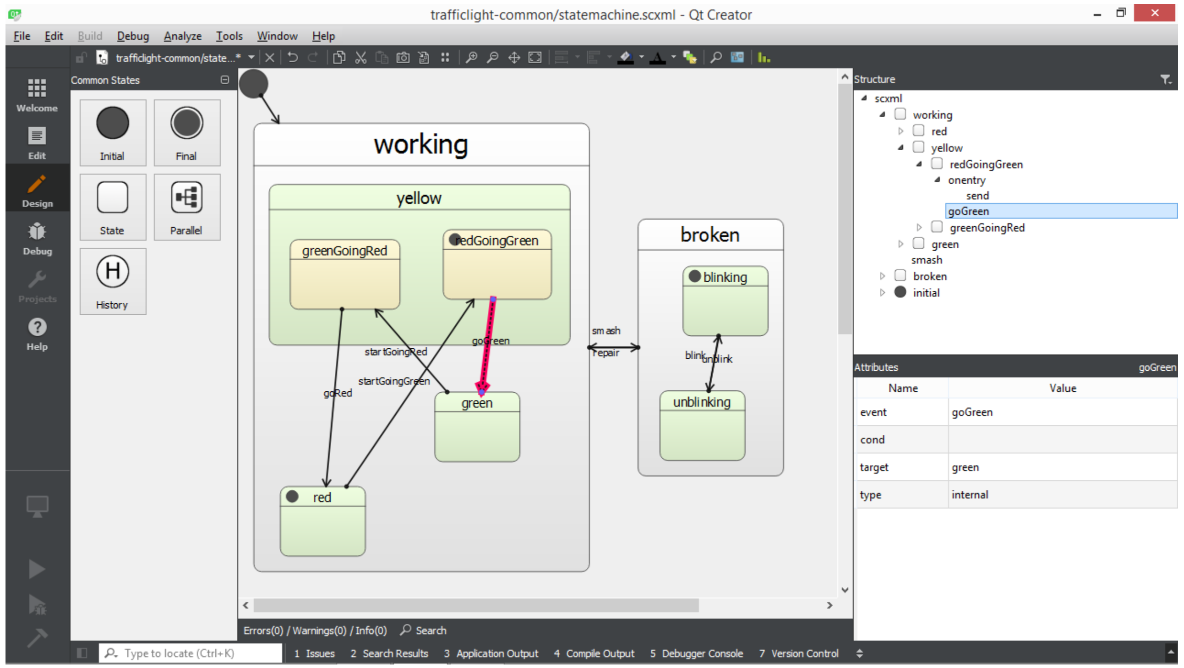
Reduce risk of unexpected system behaviour with fast & easy state chart integration

Ultimate Performance, Reliability and Stability

- Reduce risk of unexpected behavior
- Ensure maximum uptime
- Fast and easy integration
- Concise representation of business logic

The developer experience with productivity at its core

- Formalize applications & validate workflows with SCXML to define state machines
- Easily drive your state machine from a state chart rather than business logic expressed in C++ or QML
- SCXML docs can be compiled into C++ and readily integrated into any Qt application
- SCXML can also be dynamically loaded and interpreted at runtime



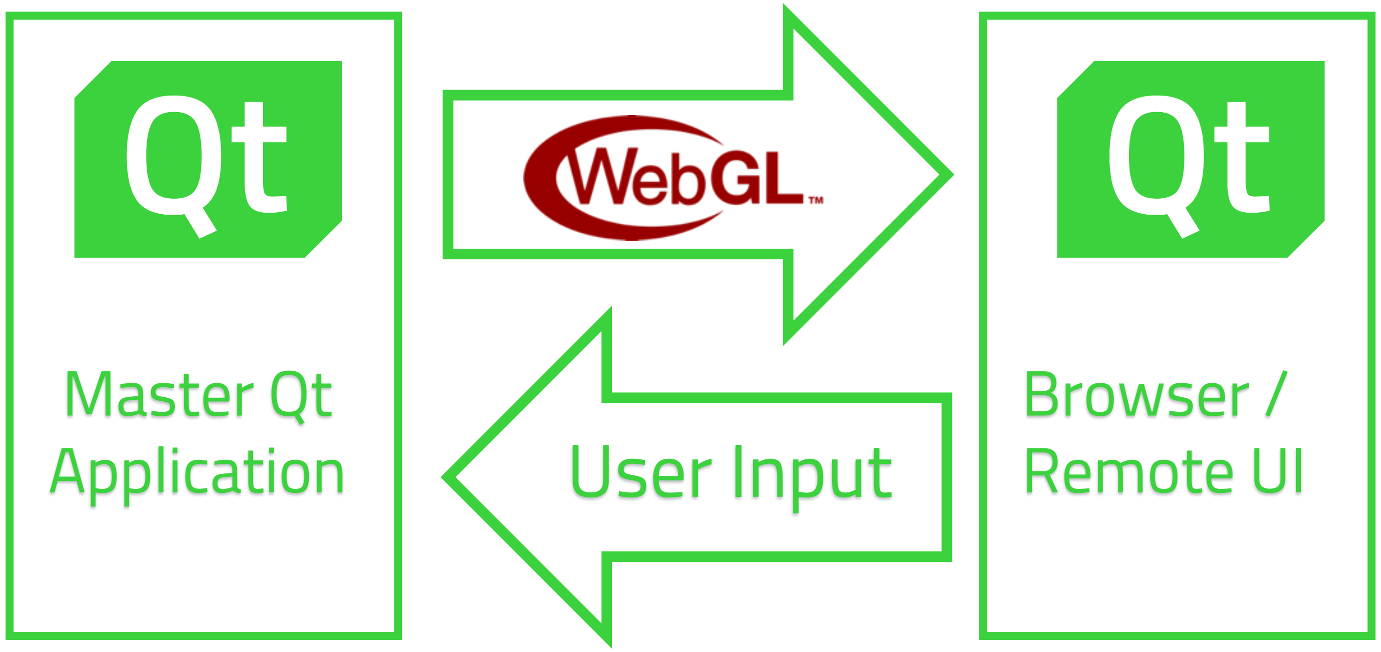
Name	Value
event	goGreen
cond	
target	green
type	internal

WebGL Streaming

- Remote control of a Qt applications with a browser
- Stream user interface and input between two Qt applications
- Think of as Android Auto / CarPlay in reverse

Remote Control Qt Apps with a Browser (WebGL Streaming)

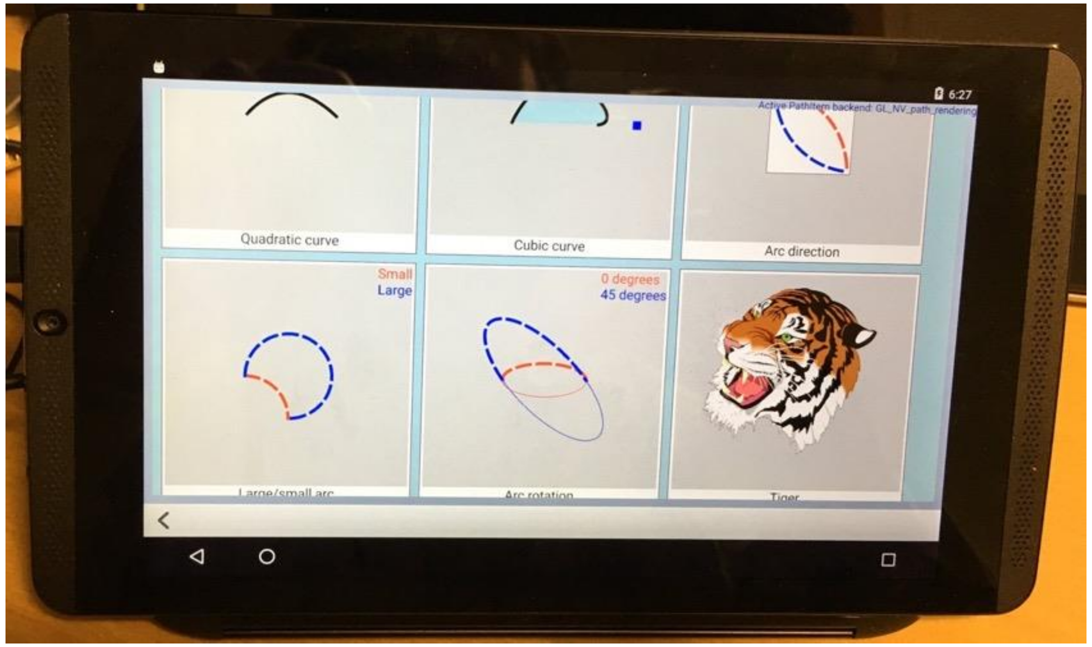
- Remote display and control of a Qt application with a common web browser or a Qt application
- Connection over a network
  - Streaming of OpenGL commands via compressed WebGL
  - User input in return channel
- Implemented as a new QPA plugin and JavaScript component that works with common web browsers (no install needed)
- Planned for Qt 5.10

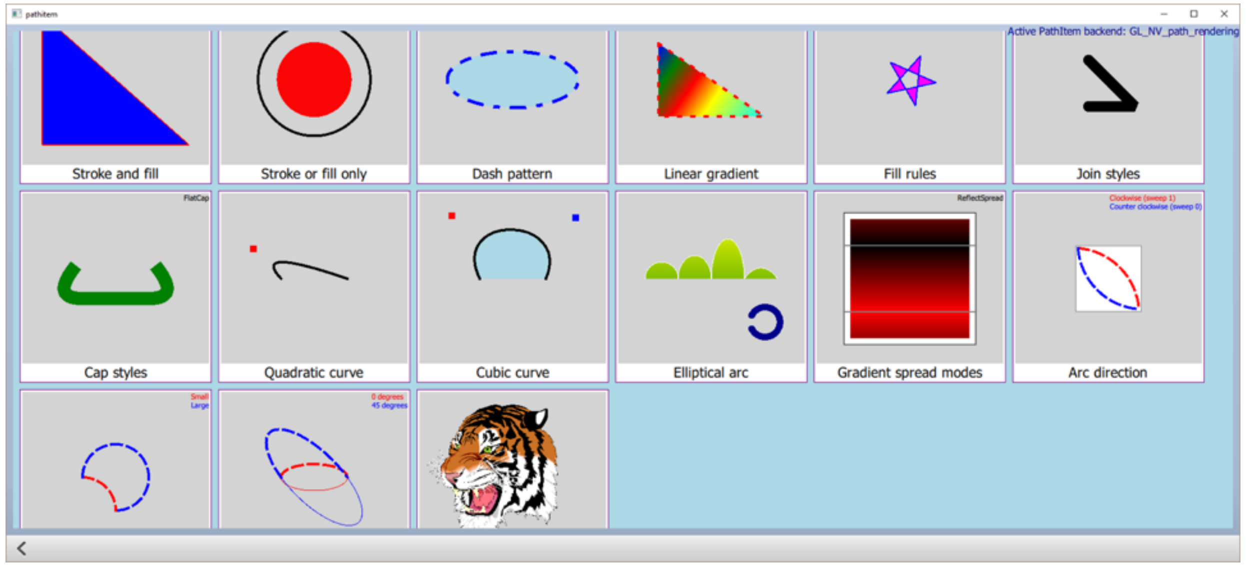


Path Rendering with Qt Quick

Path Rendering with Qt Quick

- Draw paths and shapes with Qt Quick
- Declarative way to specify complex shapes
- Significant performance increase compared to previously existing approach
- Utilizing HW acceleration
  - Works on any OpenGL HW
  - Vendor specific extensions, e.g. on NVIDIA
- Leveraging QSGRenderNode to issue native rendering commands in a scenegraph frame without going through an additional render target
- Planned for Qt 5.10



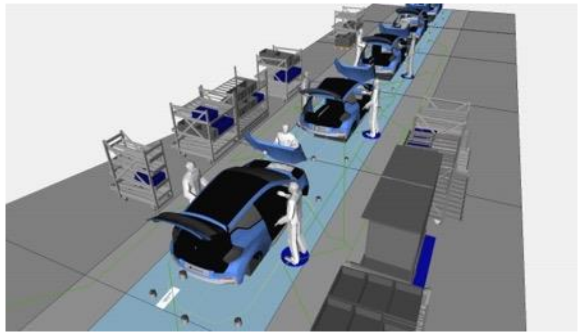


Qt 3D and Qt 3D Studio

Today: Industries Embracing 3D User Interfaces and VR

Car Clusters/IVI Systems

Medical / Learning

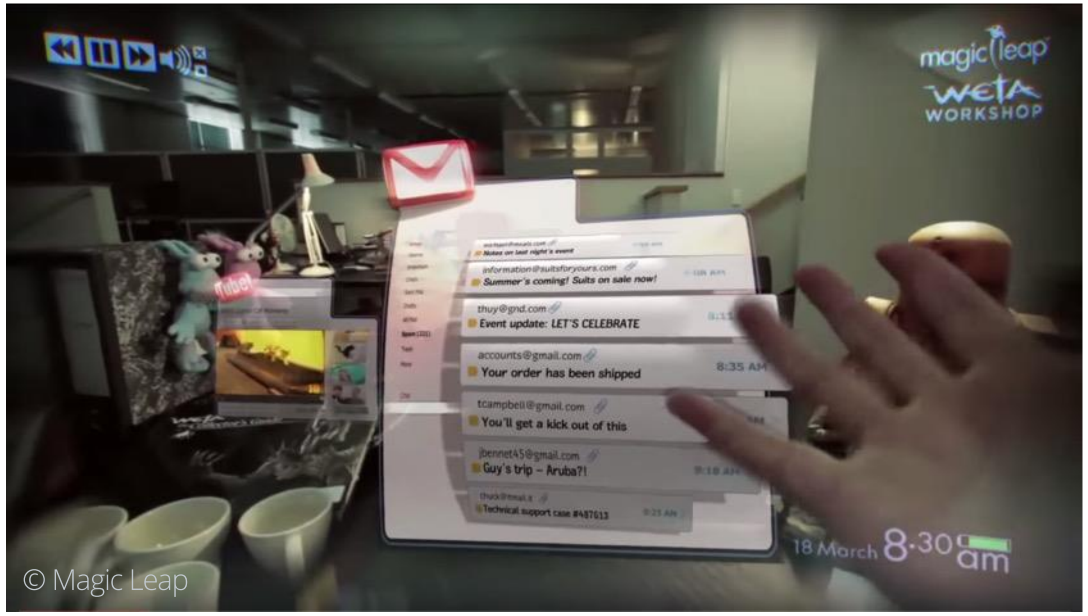

Industrial Automation

Product Visualization

3D Modeling / Animations

Virtual Reality

Tomorrow: Mixed Reality



information@suitsforyours.com
Summer's coming! Suits on sale now!
thuy@gnd.com
Event update: LET'S CELEBRATE
accounts@gmail.com
Your order has been shipped 8:35 AM
tcampbell@gmail.com
You'll get a kick out of this
jbennet45@gmail.com
Guy's trip – Aruba?! 9:18 AM
Technical support case #487613

Microsoft HoloLens

magic leap™

Building Blocks in Qt

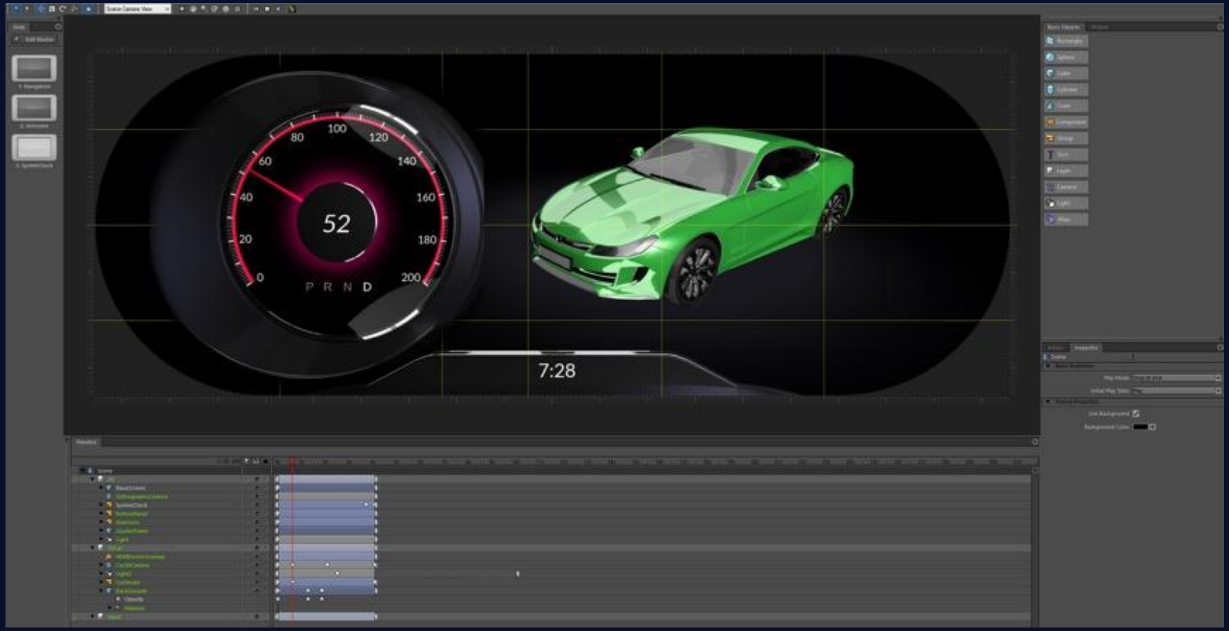
Qt 3D

- Programmer oriented
- Fully data driven 3D engine with QML API
- Runtime supports “everything”
- Very deep
  - Learning requires understanding graphics pipeline
  - Qt 3D Scene Editor helps
- Extensibility:
  - Framegraphs
  - Custom shader materials
  - Post processing shader effects

Qt 3D Studio

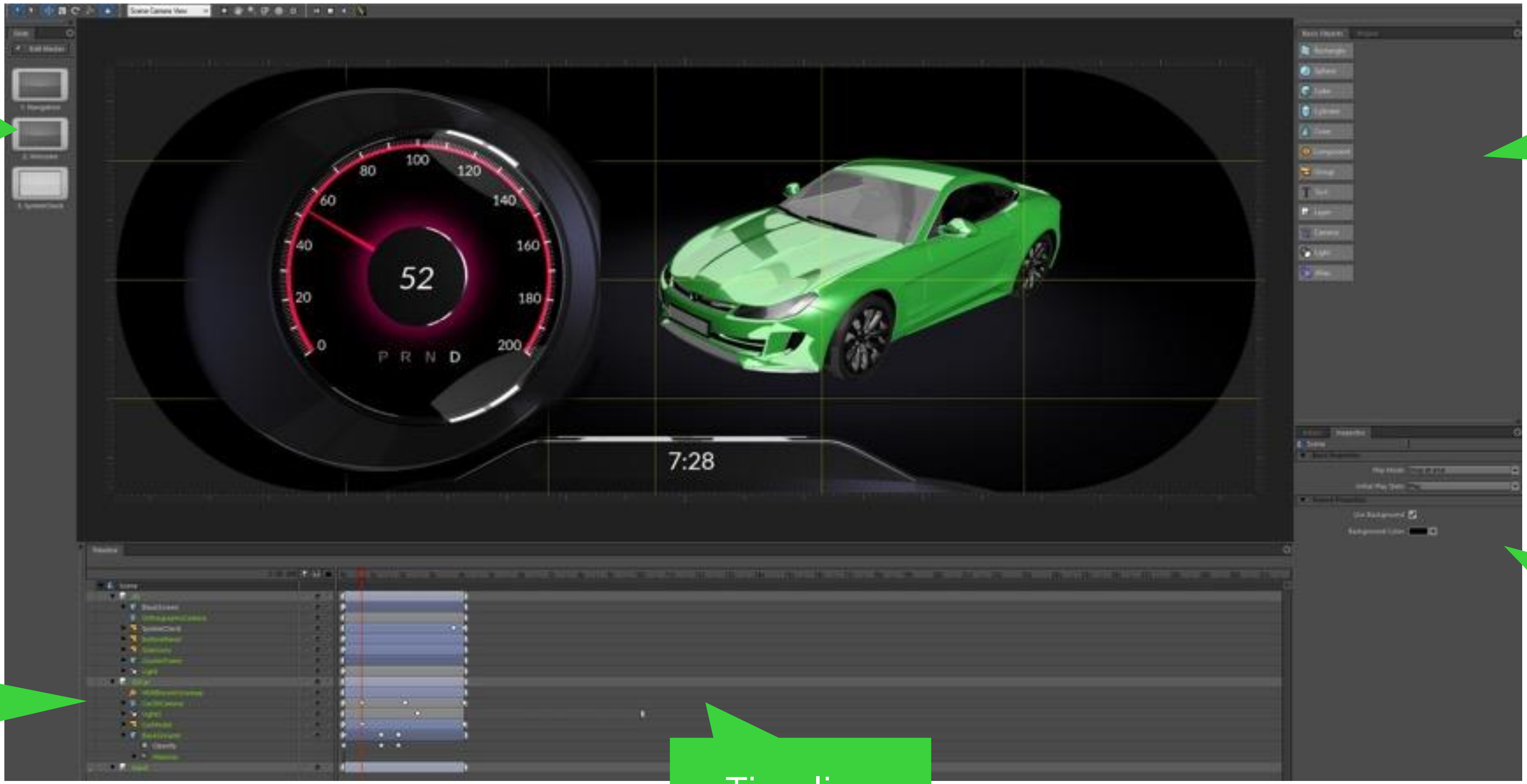
- Technical artist oriented
- Tooling that outputs data package for runtime
- Runtime supports the tooling
- Easy to approach
  - Tool can be learned in 2 days if you know how to use any 3D tool and Photoshop
  - Quick to get results.
- Extensibility:
  - Custom shader materials (integrate with UI)
  - Post processing shader effects (integrate with UI)

Qt 3D Studio



- Major code contribution from NVIDIA - NVIDIA Drive Design Studio and Rendering Engine donated to the Qt Project
- Takes Qt 3D UI tooling to a completely new level, allowing for rapid 3D UI creation and deployment.
- Goal is to fully integrate it with the Qt tooling environment
- Early access releases available, first official releases during H2/17

User Interface



Slides (UI States)

Project assets

Properties & actions

Layers

Timeline editor

3D Assets

› Import 3D geometry and animations from popular 3D Design tools (e.g. Maya, MODO and Blender) using FBX and COLLADA exchange formats

› 3D Objects and scene hierarchies are imported

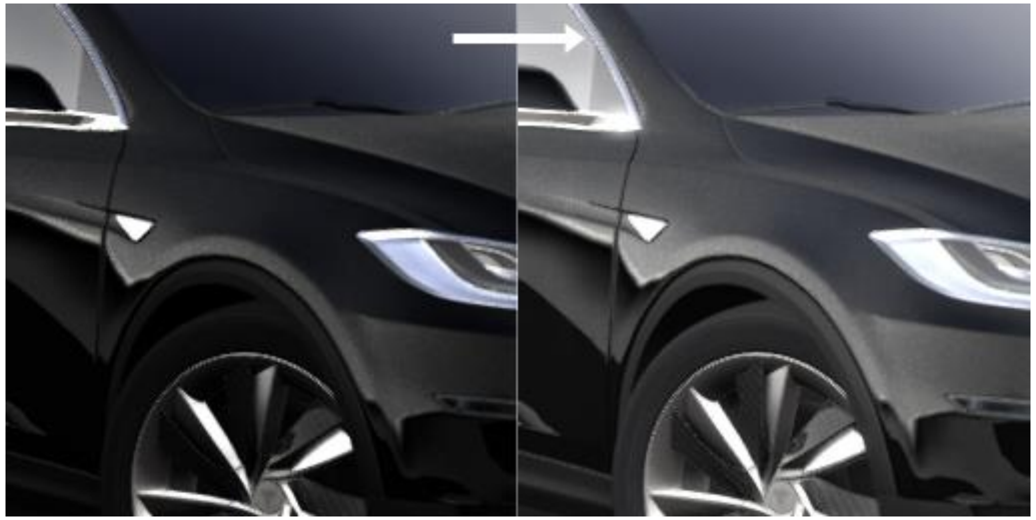
› Materials are imported and mapped to corresponding Qt 3D Studio materials

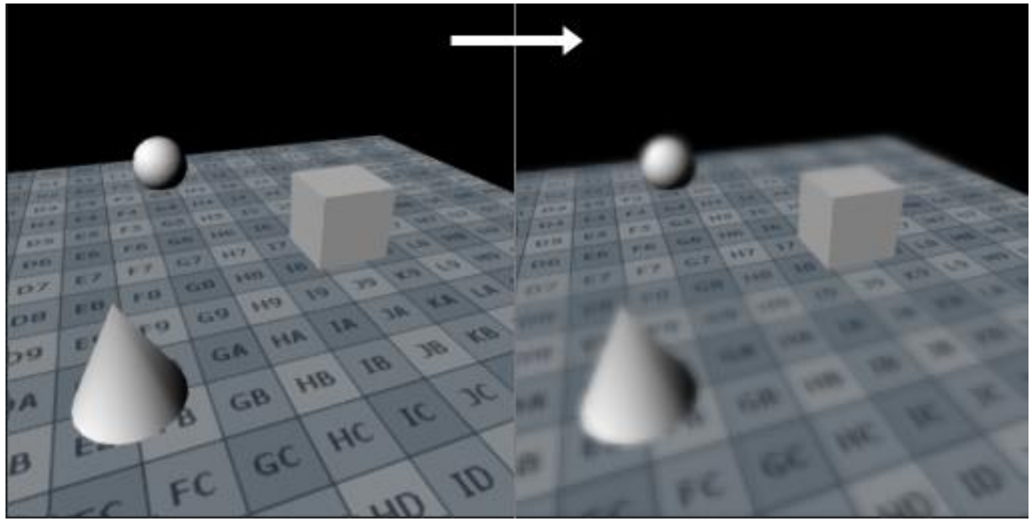
› Cameras and Lights - these objects will only come into Studio as empty nodes

› Animations - We recommend using Studio’s animation capabilities whenever practical. This helps keep mesh information on import clean and reduces conflicts between imported mesh animation and Studio’s animation upon refreshing.

Post processing effects – in built

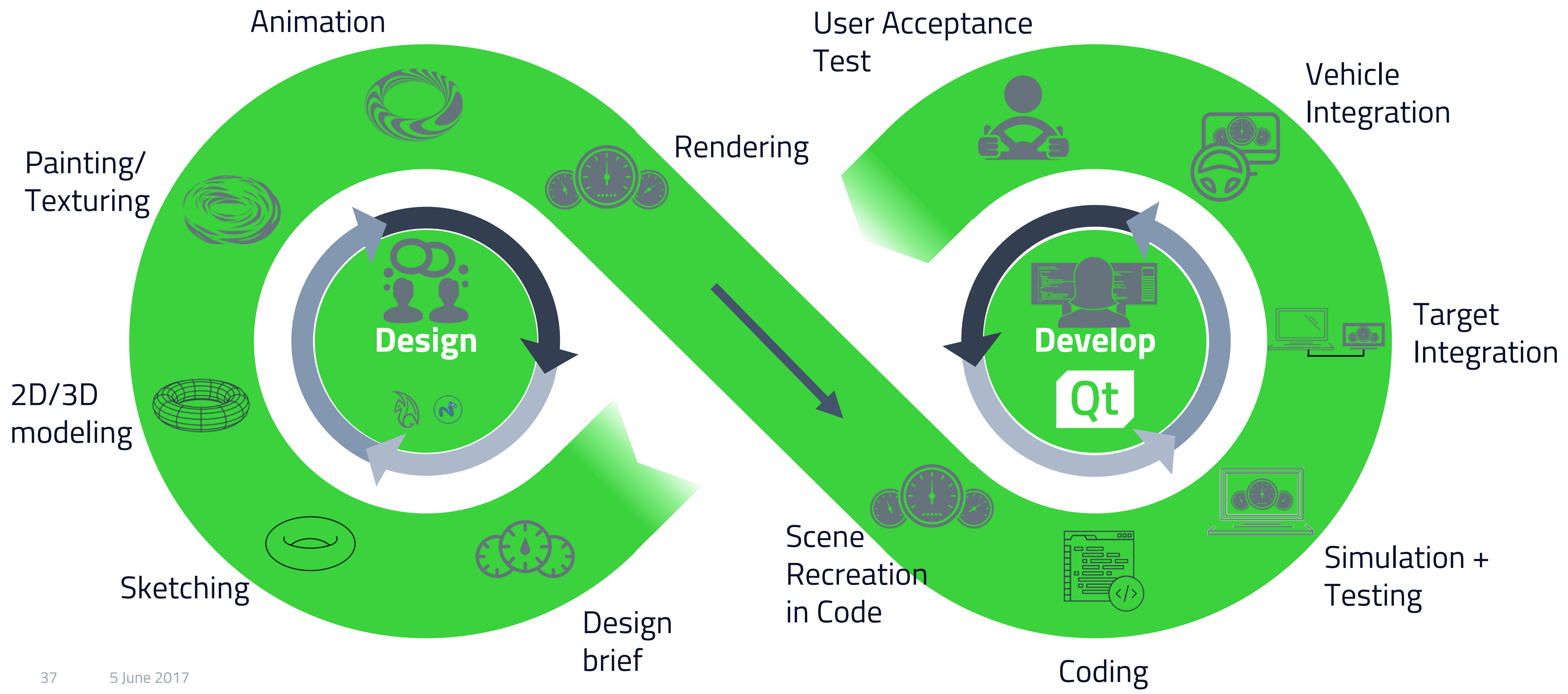
- Lens effects
  - Depth Of Field Bokeh
  - Depth Of Field HQ Blur
  - Tilt Shift
  - Chromatic Aberration
- Distortion
  - Ripple
  - Sphere
  - Spiral
- Stylize
  - Edge Detect
  - Brush Strokes
  - Corona
  - Scatter
- Blur
  - Gaussian Blur
  - Motion Blur
- Color & light
  - HDR Bloom Tonemap
  - S-Curve Tonemap
  - Sepia
  - Bloom
  - Light Table
- Antialiasing
  - SMAA
  - FXAA


HDR Bloom Tonemap

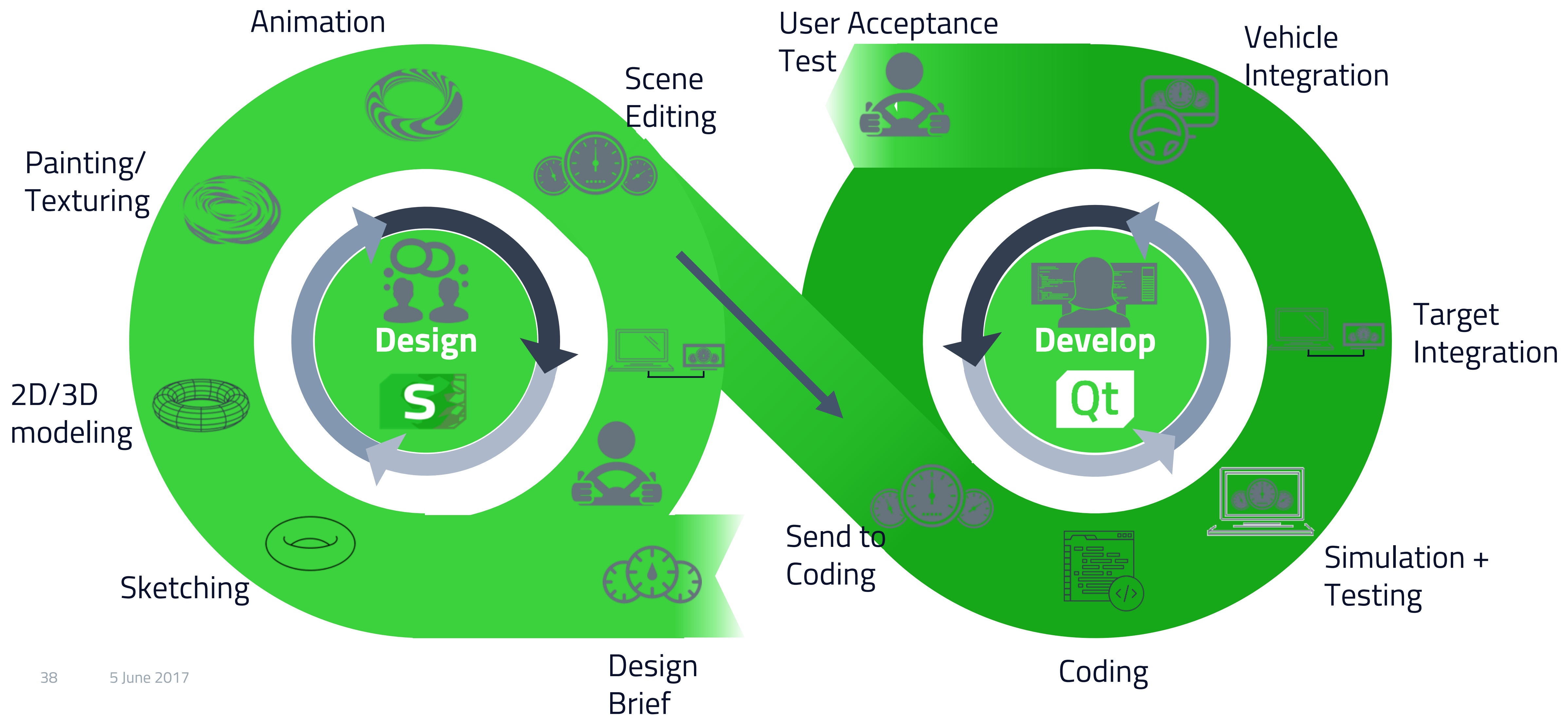

Tilt Shift

Iterative UI Development

Qt with Traditional UI Development



Iterative UI Development with Qt 3D Studio



Qt 3D Studio release roadmap 2017

January	February	March	April	May	June	July	August	September	October	November	December
			◆ ◆	◆	◆		◆		◆		◆

Internal releases

Internal binary releases for demonstrating solution to key customers & partners

Main tasks:
- Code cleanup and bug fixing
- Initial Qt integration
- Initial documentation
- 2-3 Examples/demos available

HW & OS Support
- Editor & Runtime: Desktop Windows only

Early access releases

Binary releases for key customers & partners

Main tasks:
- Code cleanup and bug fixing
- Improved Qt API
- Improving documentation
- 2 Additional demo applications

HW & OS Support
- Cross Platform Editor (Windows, Mac)
- HW: NVIDIA TX1, Intel
- Embedded Linux, Android

Official releases

General availability Commercial & Open Source

Main tasks:
- Fixing customer reported issues
- Extend embedded OS support
- Research on changing the rendering engine to Qt 3D

HW & OS Support
- Editor: Linux support
- HW: Qualcom, Renesas H3
- Integrity & QNX Support

Functional Safety for Digital Cockpit

Addressing ISO26262 Requirements with Qt

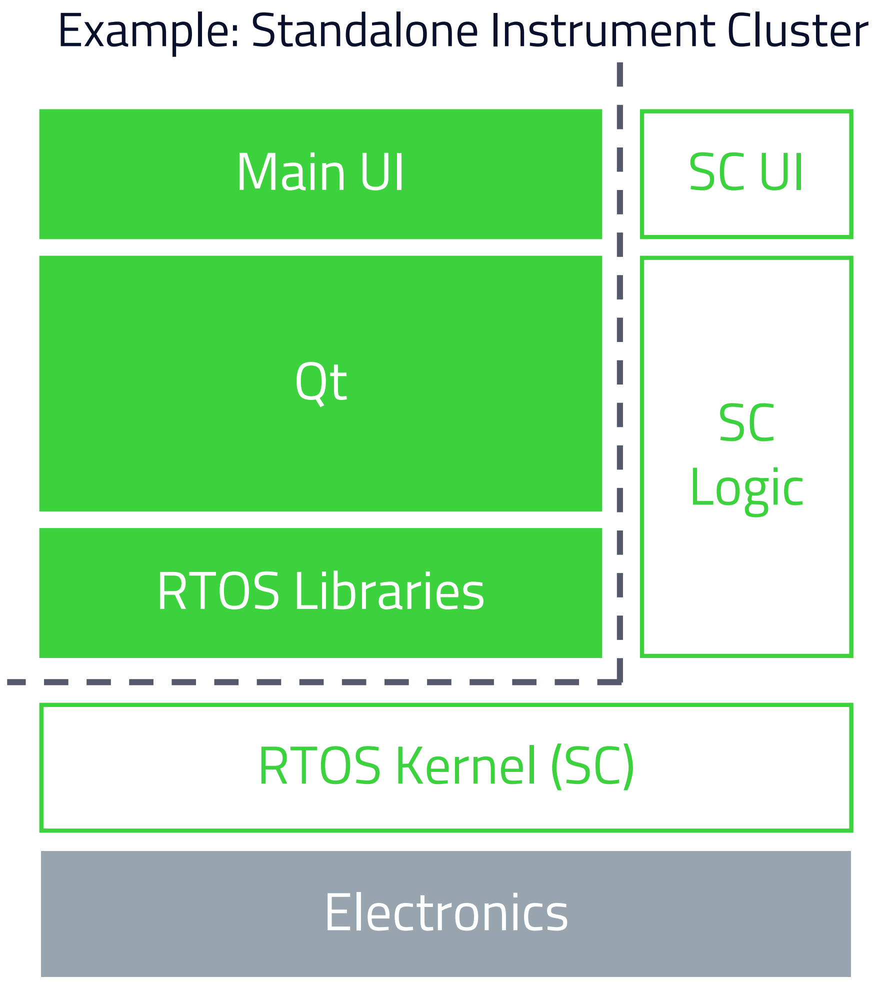
12th May 2017

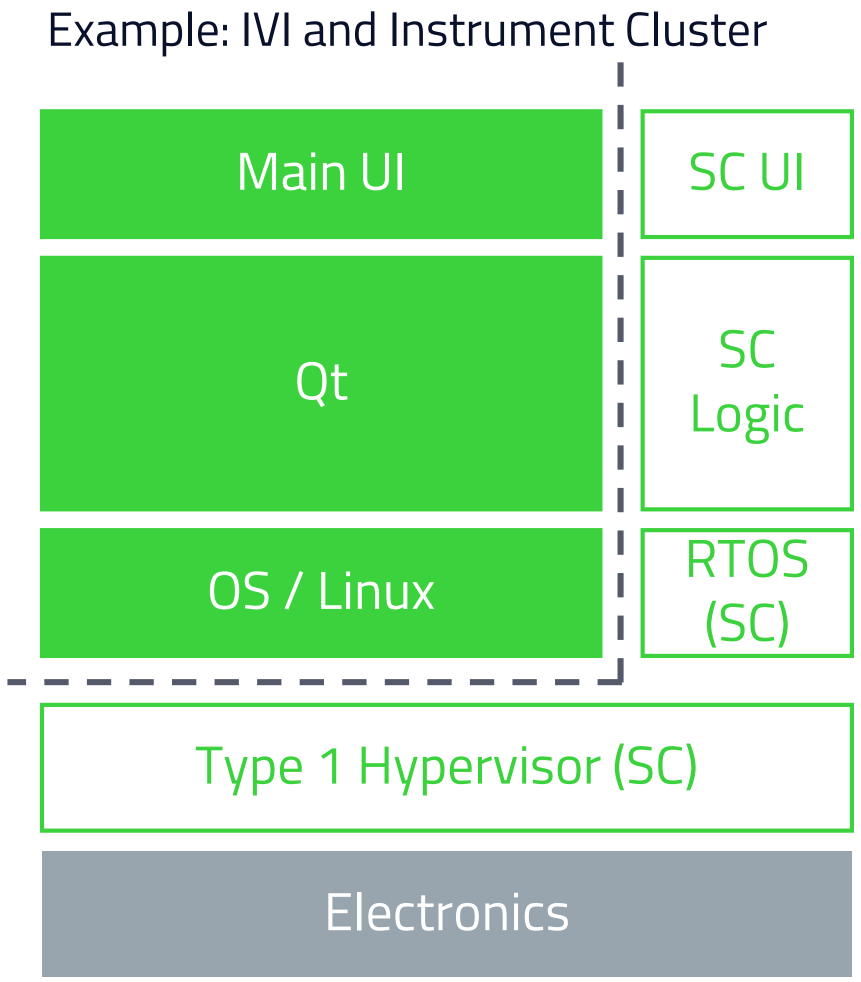
Creating a ISO 26262 ASIL B Certified System with Qt

Qt can be used in a system certified according to ISO 26262 ASIL B

- Safety critical functionality needs to be adequately separated from other functionality using a certified RTOS such as INTEGRITY or QNX
- Certified systems for multiple different industries have been created with Qt – not only automotive
  - Certification e.g. according to IEC 51608, IEC 62304 and ISO 26262
- Qt Safe Renderer component makes it easier to include safety critical parts to Qt Quick UI

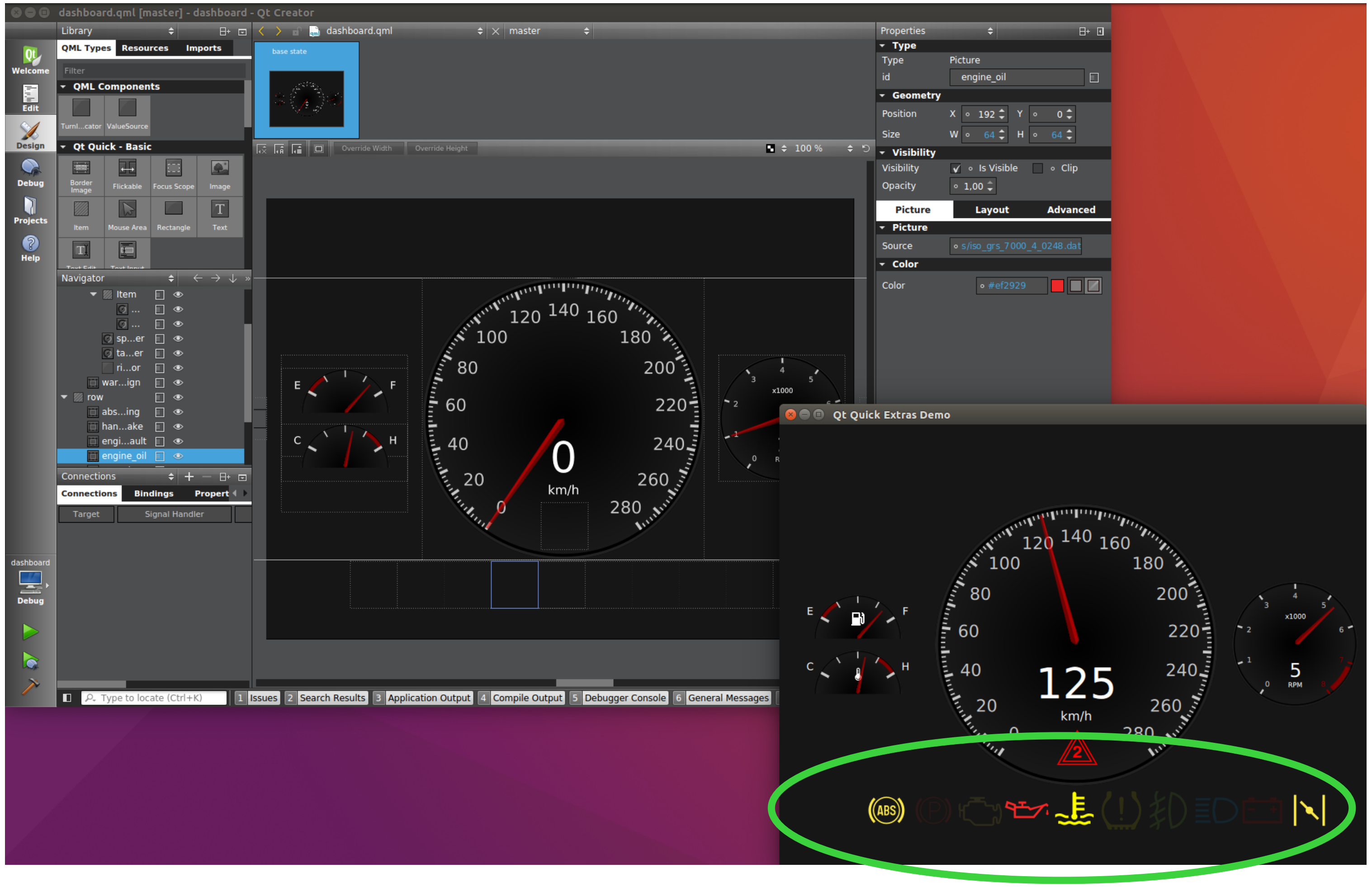
Architecture for Qt Based Digital Cockpit





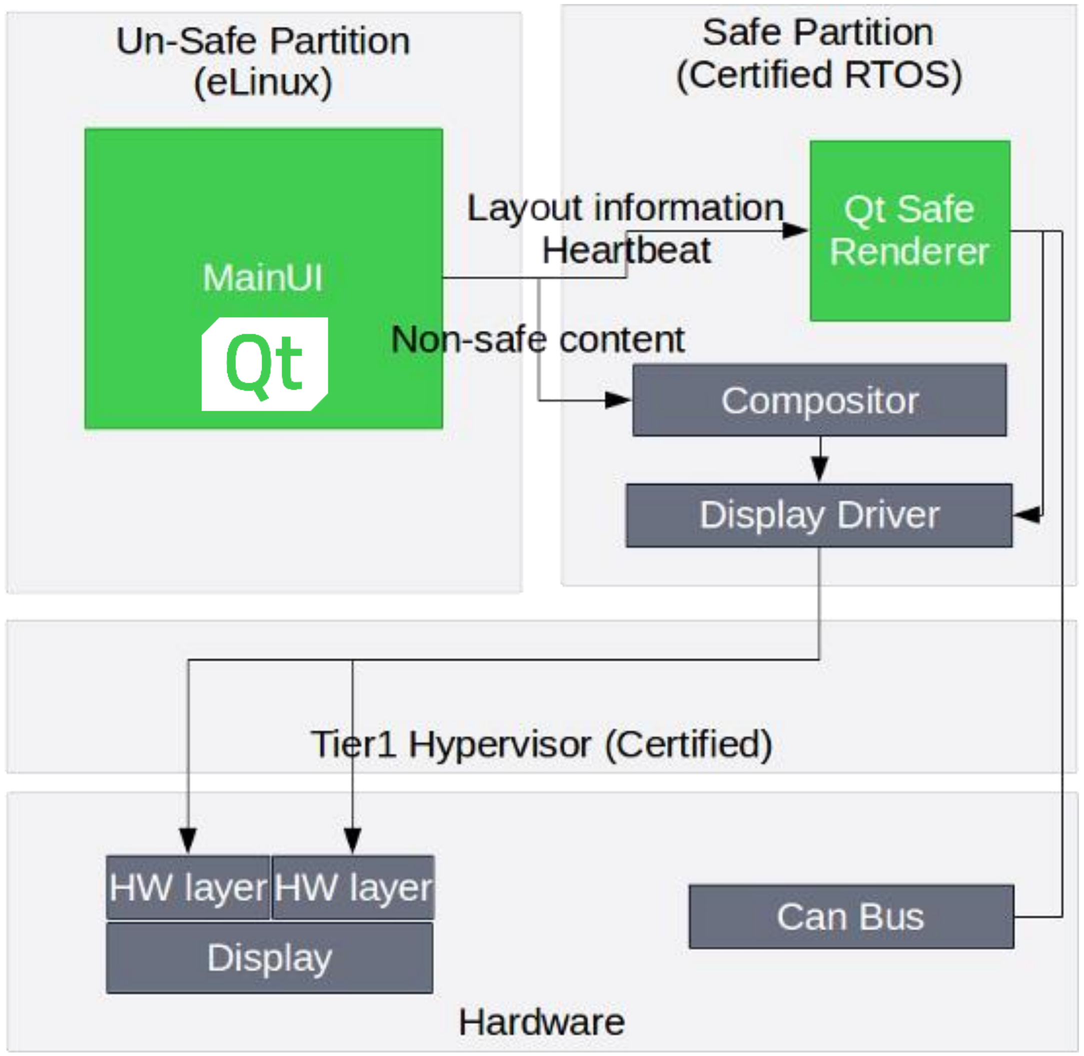
Qt Safe Renderer – Convenience for Safety Critical UI

- Certified Qt Safe Renderer
- Tooling to mark safety critical items.
- Change UI without need to modify Qt Safe Render
- Integration to Qt Quick Designer
  - Full set of ISO standard icons
  - Drag and drop safety critical item to UI
- Proof of concept based on Qt 5.9
- Certification activities planned to be completed during H2/17



Qt Safe Renderer – Architecture

- Key parts
  - Qt Safe Renderer (safety critical runtime component)
  - Integration to Qt Quick Designer and Creator IDE (safety critical tooling component)
- Integration to RTOS
  - QNX
  - INTEGRITY
- Supported HW
  - NVIDIA Drive CX
  - NXP i.MX6
  - Renesas H3
  - Qualcomm Snapdragon 820
- Qt Safe Renderer controlled via CanBUS messages during operations – no dependency to Main UI



INTEGRITY Support

- Re-introduce support for INTEGRITY with Qt 5.9
- Last supported in Qt 4.8
- Initially NXP i.MX6 and NVIDIA Tegra X1
  - Intel, Renesas, Qualcomm, … to be added in subsequent releases

Migrating code built with Open Source Licensed Qt to Production with a Commercial License

We hear your concerns

- Per our commercial license agreement, all code for commercial deployment must be developed with commercially licensed Qt.
- Can I take Qt code developed for AGL or GENIVI into a production program?
- Not if it was developed with an open source licensed Qt
- But we can always make exceptions.
- We will make a blanket exception for AGL and GENIVI.

Our Promise

- Any AGL member may use the open source licensed Qt within the AGL project, take any work or proof of concepts built within the AGL project scope using open source licensed Qt, and transfer that work into production programs by purchasing the commercial license with no penalty.
- Similarly, any GENIVI member may use the open source licensed Qt together with the GENIVI GDP, take any work or proof of concepts developed within the GENIVI GDP scope using open source licensed Qt, and transfer that work into production programs by purchasing the commercial license with no penalty.

We want AGL/GENIVI to be successful

- Any questions?
  - Ask a question now
  - Talk to us in private
  - Or send email to automotive-licensing@qt.io

Checkout our demos

- AGL homescreen demo with QtWayand
- HTML5 comparison with QML (not ours)

Thank You!

tuukka.turunen@qt.io