Meet Qt

The Leading Cross-Platform Application and UI Framework

The Qt Company

June 14 2016

The Qt Company: A Brief Introduction

- Responsible for all Qt operations globally
- Worldwide leader in
  - Qt API development
  - Device Creation and Application Development
  - Design services – UI and UX
- Trusted by over 10,000 customers worldwide
- 20+ years of Qt experience
- 200 in-house Qt experts
- Fast growing
- 27M€ revenue in year 2015

Qt is Used Everywhere

10,000+ Companies from 70+ industries use Qt

Qt in Automotive

Unify Your World with Qt

Qt Automotive Suite v1.0

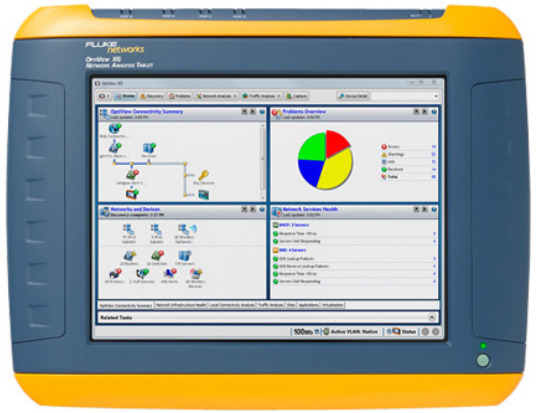
Customized for Automotive Needs

Qt.io/qt-automotive-suite/

Where There’s a User Interface, There’s Qt

Automotive IVI

Refrigerators & Coffee Machines



Network Analyzers

Plus:

- Medical Devices
- Home Automation
- Digital Photo Frames
- Set Top Boxes
- Industrial/UMPCS
- and many, many more …

The Leading C++ Cross-Platform Framework

Cross-Platform
Class Library

One Technology for All Platforms

Integrated
Development Tools

Shorter Time-to-Market

Cross-Platform
IDE, Qt Creator

Productive development environment

Used by over 1 million developers in 70+ industries

Proven & tested technology – since 1994

Qt is Used for...

Application Development

on Desktop,
Mobile and Embedded

Creating Powerful Devices

Device GUIs,
Ecosystems and whole SDKs

Target All Your End Users with One Technology

Embedded:

- Embedded Linux, Windows Embedded
- RTOS: QNX, VxWorks, INTEGRITY

Desktop:

- Windows, Linux, OS X
- Solaris, Enterprise UNIX

Mobile:

- Android, iOS, Windows Phone, Windows 10/WinRT (Windows Store Apps)

Qt Developer Offering, Cross-Platform APIs

Essentials

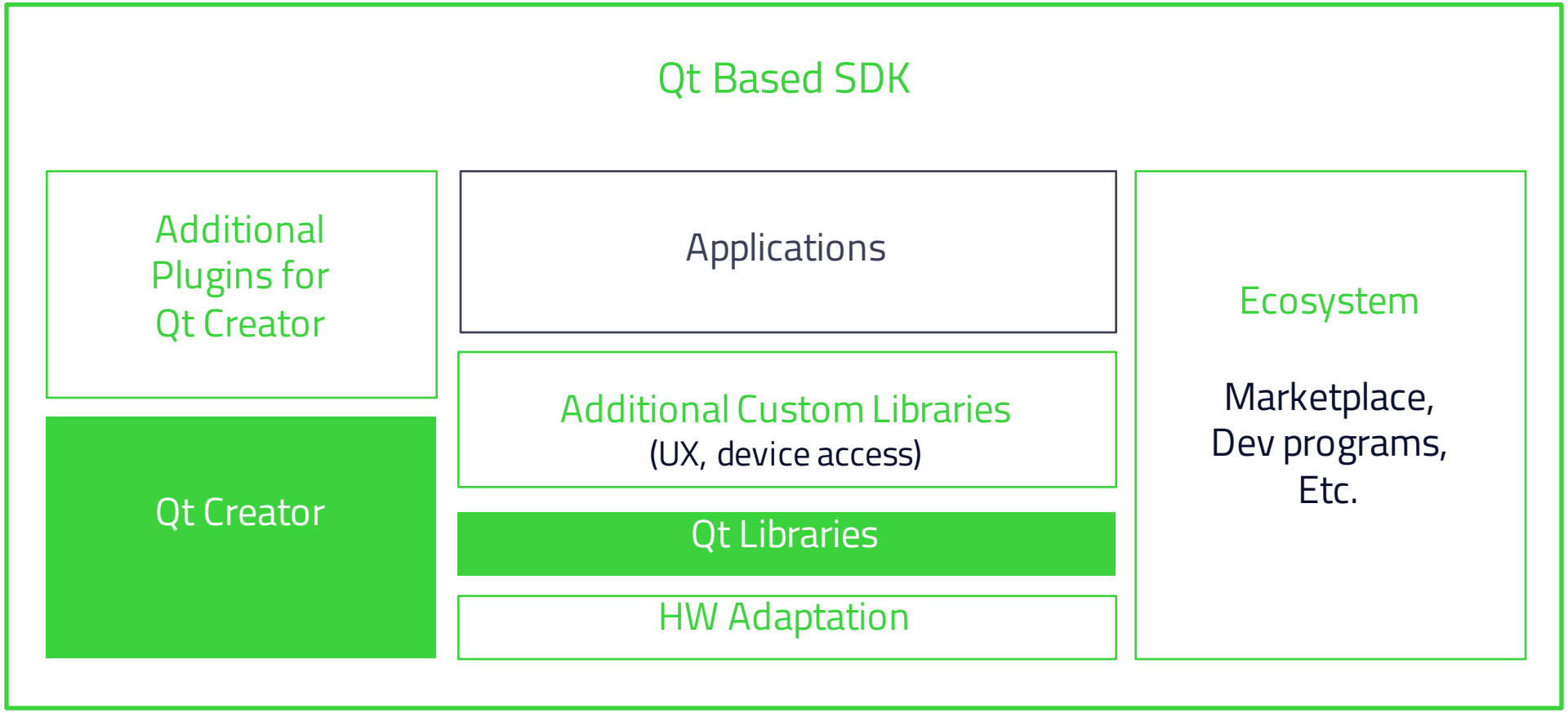
GUI	**Widgets** C++ Native LAF Layouts Styles OpenGL	**Qt Quick** QML Controls Layouts Styles OpenGL	**WebEngine + WebView** HTML 5 Hybrid UIs
non-GUI	**Core** Processes Threads IPC Containers I/O Strings Etc.	**Multimedia** Audio Video Radio Camera	**Network** HTTP FTP TCP/UDP SSL
		Sql SQL and Oracle databases	**Qt Test**

Add-ons

Charts	3D
SVG	Data Visualization
Canvas 3D	Virtual Keyboard
Serial Port	Bluetooth
Positioning	Concurrency
Printing	Scripting
NFC	Platform Extras
XML	Sensors
Image formats	In-App Purchasing

Qt Creator

Qt as Complete Technology Platform



Meet Qt 5.7

May 2016

Agenda

- Redefine User Experience
  - New Dimensions – Qt and 3D
  - Productivity out-of-the-box: Qt Quick Controls 2.0
  - Functionality meets design – Qt Quick Designer
- The Framework for Modern C++
- Get Ahead of the Rest! Shorter time-to-market for embedded devices
- New Licensing Offering
- Summary of Qt 5.7 Highlights

Redefine User Experience

- New dimensions
- Productivity Out-of-the-Box
- Functionality Meets Design

Pioneer in User Experience Creation

- For the past 20 years, Qt has kept pace with market demands for UI creation to create the best UX for your end users
  - Classic desktop look-and-feel
  - Modern touch-based embedded screens
  - Personalized mobile applications
  - True multi-screen user experience

- Qt gives You
  - Multiple approaches for UI creation to match your needs
  - Full native performance, leveraging OpenGL and hardware acceleration where possible
  - Declarative design language with Qt Quick for easy developer-designer workflow
  - Hybrid HTML5 integration, full browser engine through Qt WebEngine
  - Fun, productiveness, and focus on content!

New Dimensions – Qt Offering for 3D Graphics

Qt 3D

New Qt module for 2D and 3D rendering with a framework for near-realtime simulations *(e.g. physics, audio, AI, collision detection)*

Qt Canvas 3D

Use JavaScript and JS-based 3D libraries with Qt Quick

Qt and OpenGL

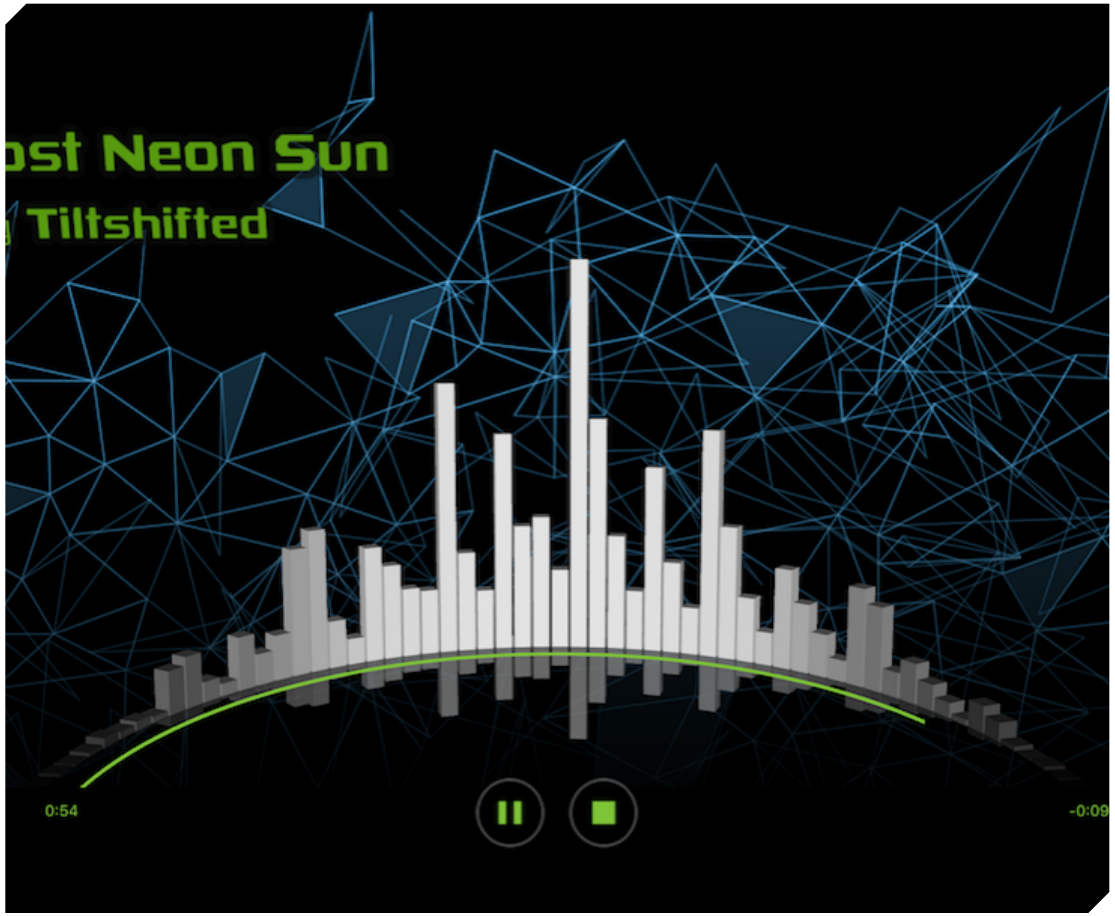
Mix and Match Qt with raw OpenGL to the maximum of your liking

Qt Data Visualization

Library for 3D charting and data visualization

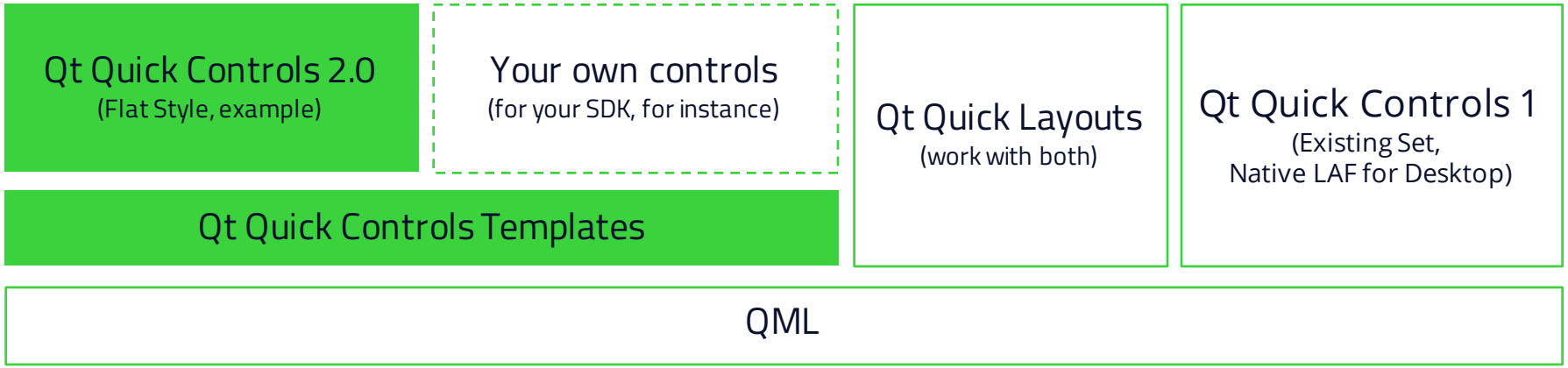
Introducing Qt 3D – Fully Supported with Qt 5.7

- 3D framework specifically tailored for Qt/QML
  - Renderer
  - Generic framework for near-realtime simulations
- Multithreaded and extensible architecture
- Split into core and *aspects* (physics, audio, collision, AI, path finding, etc)
- 3D object loaders for popular formats
- Developed together with **KDAB**, a Qt Service Partner

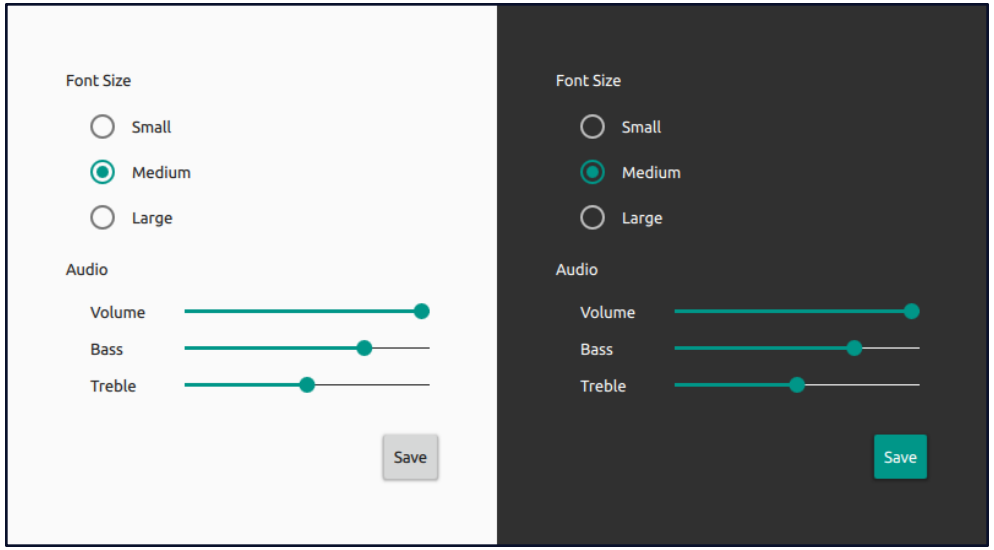


Productivity Out-of-the-Box – Qt Quick Controls 2.0

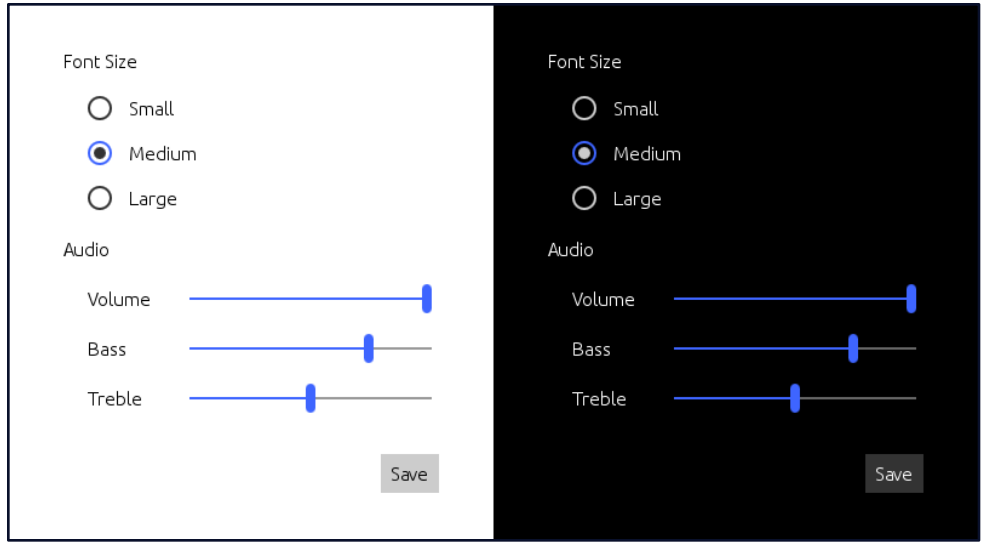
- Library of UI controls *(buttons, sliders, dials, etc.)* for Qt Quick
  - A new project, re-thinking the controls, mainly from Embedded perspective
- Sleak, performant, easily customizable, also for SDK creators
- Divided into Templates (basis) and an existing set of Controls (example set, using Flat Style)



Qt Quick Controls 2.0 - Examples



Google Material Design



Microsoft Universal Design

The Framework for Modern C++

The Framework for Modern C++

- Qt is the framework for all C++ development, following the modern C++ progress closely
  - Fully harness the power of C++ with the convenience of Qt libraries
- Qt supports C++11/C++14 features and Qt 5.7 leverages C++11 also within the API design
- Qt 5.7 requires compilers to support C++11
  - Dropping out support for older C++98 compilers
- Qt 5.6 (LTS) is a valid, parallel product for older compilers for multiple years
  - Qt 5.7+ will integrate more tightly into modern C++ features

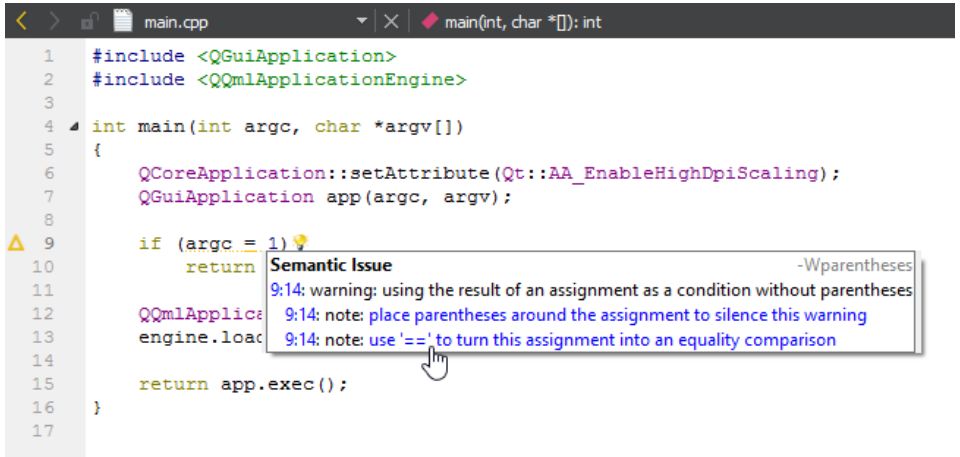
Get Ahead of The Rest

- Shorter Time-to-Market with Qt Tooling
  - Qt Creator 4.0
  - Full Embedded Tooling
  - Pre-built Software Stack

Qt Creator 4.0

- Full cross-platform development environment for desktop, mobile and embedded
  - Optimal for Qt, QML and C++ projects
  - Develop, design, deploy, test, analyze and optimize—all in the same seamless workflow!

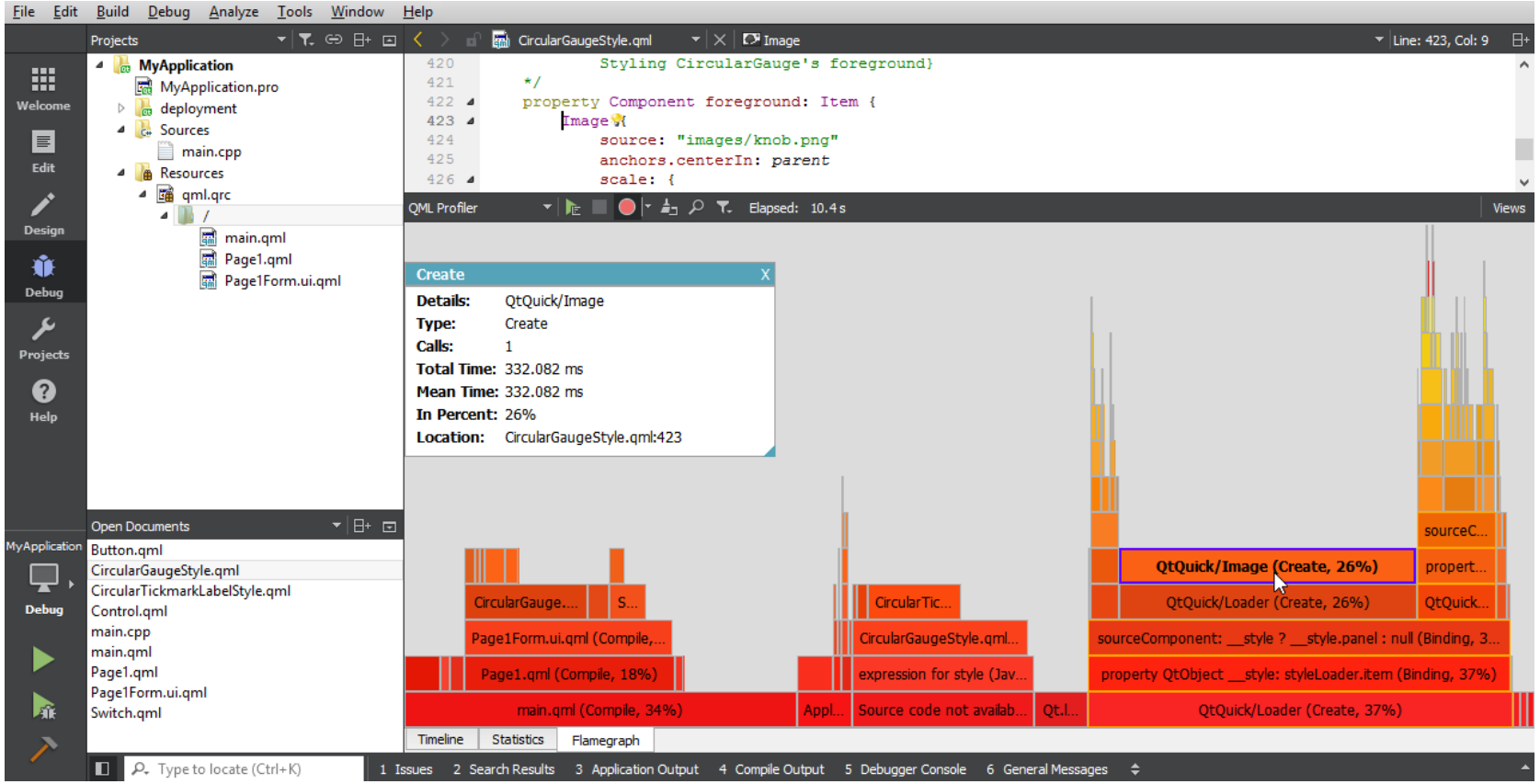
- New for Qt Creator 4.0
  - CLang Static Analyzer integration – Find problems easily in C, C++ and Objective-C programs
  - Autotest integration – Easily run autotests from your projects
  - Extended QML profiler – Analyze pixmap cache usage, scene graph performance, JS memory usage and input events
  - Improved workflow for CMake projects
  - New styles



Find problems early with Qt Creator and CLang static analyzer

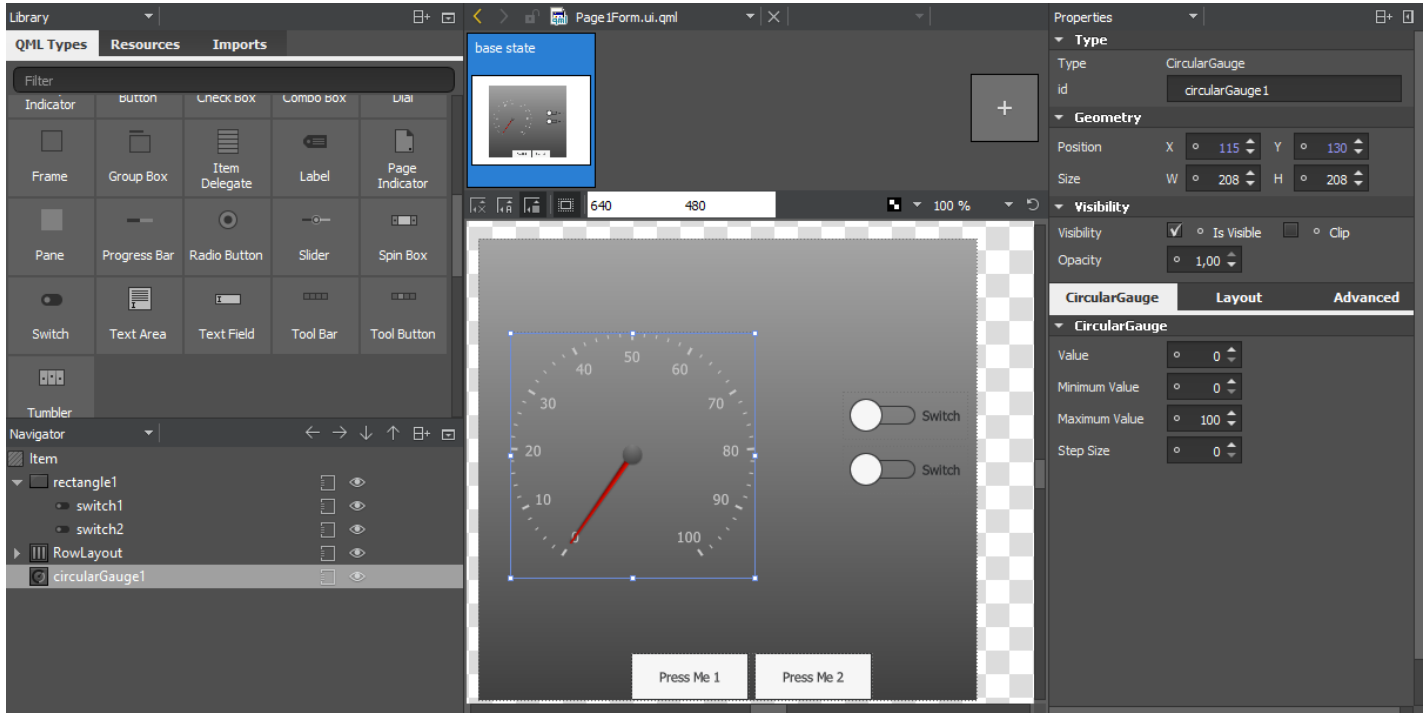
Qt Creator 4.0 – QML Profiler

Visual Analyzer for Optimization – Easiest Way to Find Causes of Your Performance Issues!



Functionality Meets Design – Qt Quick Designer

- Visual drag'n'drop UI editor
  - Built-in to Qt Creator 4.0
- Together with integrated Qt Quick Controls 2.0 provide a rapid way for UI design
  - Seamless designer-developer workflow
- Separated UI presentation (UI Forms, ui.qml files) and UI logic (regular QML files).
- A lot of work has been put into improving the designer in the past versions



Drag'n'drop all Qt Quick Controls, manage their hierarchy , layout, properties and directly connect them to each other.

Embedded Tooling

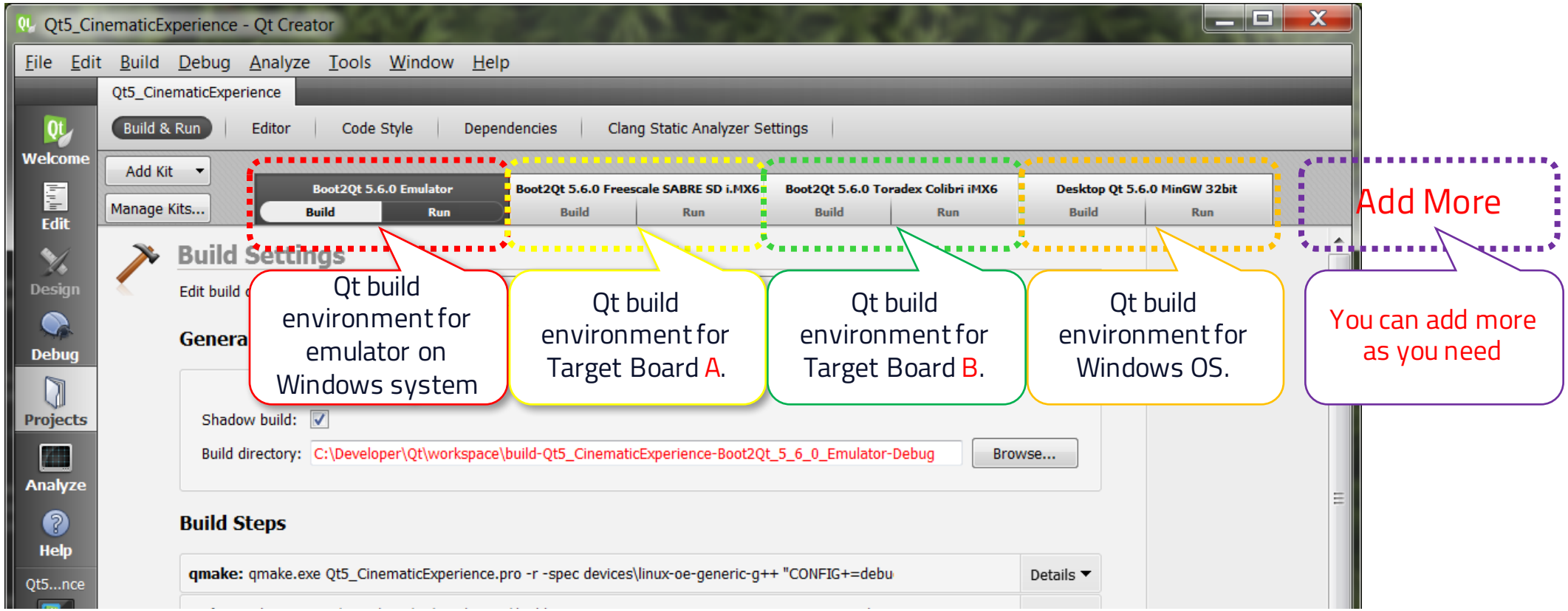
- With Qt tooling embedded development workflow is as effortless as desktop or mobile development
- Qt Creator IDE allows you to
  - Do UI prototyping with rapid design-develop-deploy cycles
  - Immediately see your software run on real embedded hardware—with one-click deployment!
  - Emulate the software without the actual HW with customizable environment and sensor simulation
  - Do embedded Linux development also from Windows host computer

Boot to Qt Software Stack

- Immediate Embedded Prototyping
- Kick-start to Embedded Projects
  - Pre-built binaries for common development boards
- Full Customization through the Yocto Project tooling

Boot to Qt

One Click Build-Deploy-and-Run to Multiple Build Environmentes



Tech Previews with Qt 5.7

- **Qt Wayland Compositor**, multi-process support for embedded
- **Qt SCXML**, state chart framework integration
- **Qt Gamepad**, a plugin-based Qt API for interfacing with gamepads
- **Qt Serial Bus**, for device bus communication, with CAN bus and ModBus implementation

Summary

Qt 5.7 Highlights

Qt 5.7 Highlights

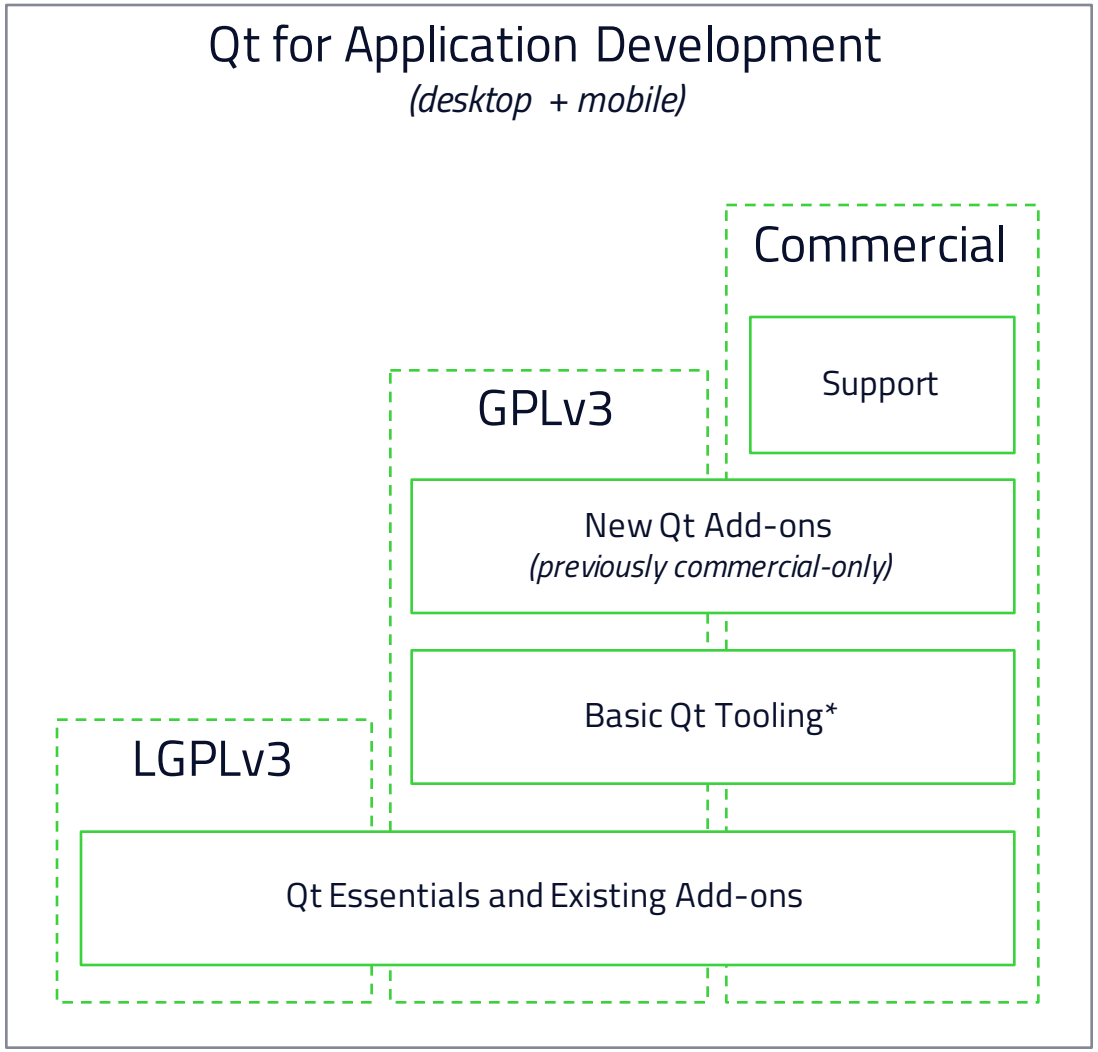
- **Qt Quick Controls 2.0** – A new and performant library of UI controls designed for embedded and mobile UIs
- **Qt 3D** fully supported
- **Qt Creator 4.0**
- Qt 5.7 is fully leveraging **C++11**, supporting the use of it and using it internally as well
  - Qt 5.7 does not support for older non-C++11 compilers (Qt 5.6 LTS supports)
- Tech Previews
  - **Qt Wayland Compositor**, multi-process support for embedded
  - **Qt SCXML**, state chart framework integration
  - **Qt Gamepad**, a plugin-based Qt API for interfacing with gamepads
  - **Qt Serial Bus**, for device bus communication, with CAN bus and ModBus implementation
- **New Licensing Terms**
  - Upgraded from LGPLv2.1 to **LGPLv3** for Open Source Qt
  - **Open-sourced** new components under GPLv3
    - **Qt Charts, Qt Data Visualization, Qt Virtual Keyboard, Qt Quick 2D Renderer, Qt Purchasing**

New Licensing Terms

- LGPL Licensing Updates
- Additional Licensing Options for Qt value-add modules
- Start-Up Licensing Tier

Updated Open Source Licensing

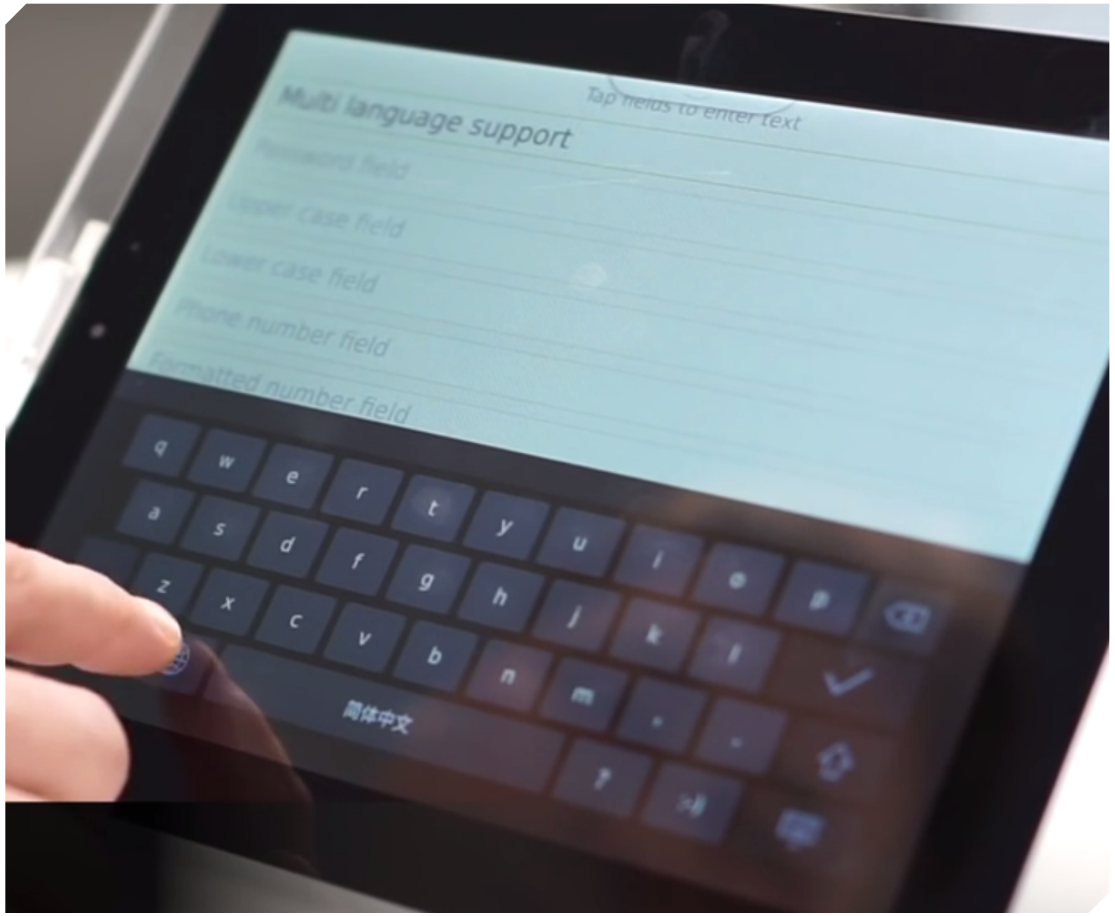
- With Qt 5.7 we are updating the open source licensing offering for Qt and harmonizing the developer offering for "Qt for Application Development" (desktop/mobile)
- With Qt 5.7 the licensing offering of Qt is
  - Application Development libraries and tools:
    - **Commercial or GPL** for some Qt Add-on libraries and all tools
    - **Commercial or LGPLv3** (or GPL) for all Qt Essential and most Qt Add-on modules
  - Qt for Device Creation remains a commercially-licensed product
    - Qt source codes available for embedded use under GPL or LGPLV3 licenses
    - Embedded-specific tooling and solutions commercial-only
- LGPLV2.1 is removed from the licensing offering
  - Note: Qt 5.6 and earlier versions unchanged



*GPLv3 tooling can be used with the LGPLv3 licensed Qt libraries

Open Sourcing Value-Add Components

- With the licensing upgrade, we have open-sourced a lot of formerly-closed libraries and tools **under GPL**, unifying the developer offering
  - **Qt Charts** – Set of easy-to-use charting components, both for static and dynamic charts
  - **Qt Data Visualization** – module for 3D data visualization and charting
  - **Qt Virtual Keyboard** – A full virtual keyboard solution with custom keyboard layouts, themes, multiple languages and handwriting recognition
  - **Qt Quick 2D Renderer** – Use Qt Quick without OpenGL support, for instance in lower-level embedded devices
  - **Qt Purchasing** – Cross-platform in-app purchasing API. Available under LGPLv3 licensing.



Qt Virtual Keyboard

Start-Up Licensing Tier

Qt for Application Development for Small Companies

- This spring, we also introduced a new low-price licensing option for start-ups and small companies
  - Starting from $79/month
  - To be eligible: annual sales revenue under $100,000

- Easy access to full Qt for Application Development product
  - Cross-platform application development for all desktop and mobile platforms
- Change and rebuild the Qt libraries. Deploy as you want to.

- Self-service subscription license through Qt Webshop under www.qt.io
- See www.qt.io/start-up-plan

Questions? Let us know and Take Qt 5.7 on a test drive

Qt.io/contact-us/

Qt.io/download/

Last but not least...

Qt World Summit

Pier 27

Cruise Ship Terminal
San Francisco, CA, USA

Attendees from Top Global Companies

Attendees from The Automotive Industry

DAIMLER

BMW

Ford

OpenCar

here

Panasonic

HYUNDAI

FORTECH
Innovative Formulations

MAGNETI MARELLI

MITSUBISHI

HELLA

Continental

Early bird sales ends July 15

2-day: $517 - regular: $690

3-day: $740 - regular: $987

Register today www.qtworldsummit.com

Thank You!

Test drive Qt 5.7 www.qt.io/download/

Get your #QtWS16 early bird pass www.qtworldsummit.com

Questions?

Jake.Petroules@qt.io

Marty.Udisches@qt.io

Sneak Peek into Qt 5.8

Overview of 2015-2016 Qt Releases

Qt 5.5 (June 2015)

-> Quality & maturity focus
-> Qt 3D TP
-> Qt Location TP
-> Unified online installer
-> New product structure
-> VxWorks with Qt 5.5.1

Qt 5.7 (June 2016)

-> License change
-> Leveraging C++11
-> Qt 3D
-> Qt Quick Controls 2
-> Yocto compliancy
-> New Tech Previews: Qt Wayland Compositor, State machine tooling (SCXML), Qt SerialBus, OTA
-> Base of Qt Automotive

Q1/15 Q2/15 Q3/15 Q4/15 Q1/16 Q2/16 Q3/16 Q4/16

Qt 5.6 (March 2016)

-> LTS release
-> HighDPI
-> Full Windows 10 support
-> New CI System
-> Webkit and Qt Quick 1 removed
-> Preview of new modules

Qt 5.8 (Q4/2016)

-> Qt Lite (configurability and optimization of resource usage)
-> Built-in Qt Quick Compiler
-> Vulkan / DirectX 12 enablers
-> Input handling renewal
-> Qt Wayland Compositor
-> Qt SCXML

Next Generation Graphics

- Research ongoing to better address the new graphics APIs
  - Vulkan, Direct 3D 12, Metal
- Remove hard dependency on OpenGL from Qt SceneGraph and to create new backends for different graphics APIs
- Target is to have something already with Qt 5.8 as experimental
  - This work is also setting the baseline for Qt 6 graphics
- On Windows there are problems with ANGLE and good OpenGL support is still an issue
  - The new Direct 3D backend may allow us to remove ANGLE dependency (perhaps with Qt 5.9)
- New approach enables improved Qt Quick 2D renderer (without OpenGL stubs)
- Qt 3D is already built to allow usage of Vulkan etc – benefits especially with complex models

Built-in Qt Quick Compiler

- Qt Quick Compiler removes the need to dynamically load the UI files by pre-compiling the QML files to the application
- Key benefits:
  - Cache for fast re-execution of JIT'ed code
  - Faster application startup
  - Improved performance on platforms that do not allow JIT
  - IPR protection by making reverse engineering difficult (not possible to get the QML code from the application binary)
- Currently a separate component, now to be built inside the Qt Quick Engine
  - Performance expected to be on the same level as the current Qt Quick Compiler
- Two possible use cases
  - Runtime mode: Storage of just-in-time compiled code on disk making second run faster
  - Build time mode: Compiling QML to C++ during application build making also first run faster
- Currently available separate Qt Quick Compiler remains a commercial-only item for users of earlier Qt versions (< Qt 5.8)

Qt Lite

Configurability and Optimization of Resource Consumption

- Minimum hardware requirements of a Qt 5 based application are sometimes unnecessarily high due to:
  - Difficulty in configuring the unneeded parts out (despite the modularization of Qt 5)
  - Use of resources in some areas of Qt is not optimized for low end devices
  - Lack of guidelines, recommendations and HW requirements for creating a system wit limited resources
- Qt Lite project is aiming to increase the configurability of Qt 5
  - Being able to easily optimize Qt better for various use cases and to configure the set of used APIs
- Additionally effort is put especially into optimization of resource consumption without loss of functionality
- For some parts an alternative solution for low resource consumption may be created (with reduced functionality, but still meeting the most likely use cases, for example ICU)
- Work with customer use cases in order to optimize maximally and to be able to better communicate the system requirements for different use cases

Qt Lite

Requirements and Possible Use Cases

- Targeting to have basics set for Qt 5.8 – but work will continue in subsequent versions
- Qt Lite is not a fork of Qt, it provides the same developer API, uses the same codebase, infrastructure and tools
- Qt Quick as UI layer, other key modules Qt Core, Qt GUI, Qt Network optimized for resource consumption
- Supports both GPU and non-GPU HW
- Suitable CPU e.g. ARM Cortex A5 with 300MHz
  - Looking into possibility of using Cortex M7
- Well feasible to run a system with 32MB RAM using embedded Linux and main Qt functionality
- Minimal configuration, amount of RAM and ROM depend on the application
- Smaller footprint benefits also boot time and power consumption – in addition to lower HW cost

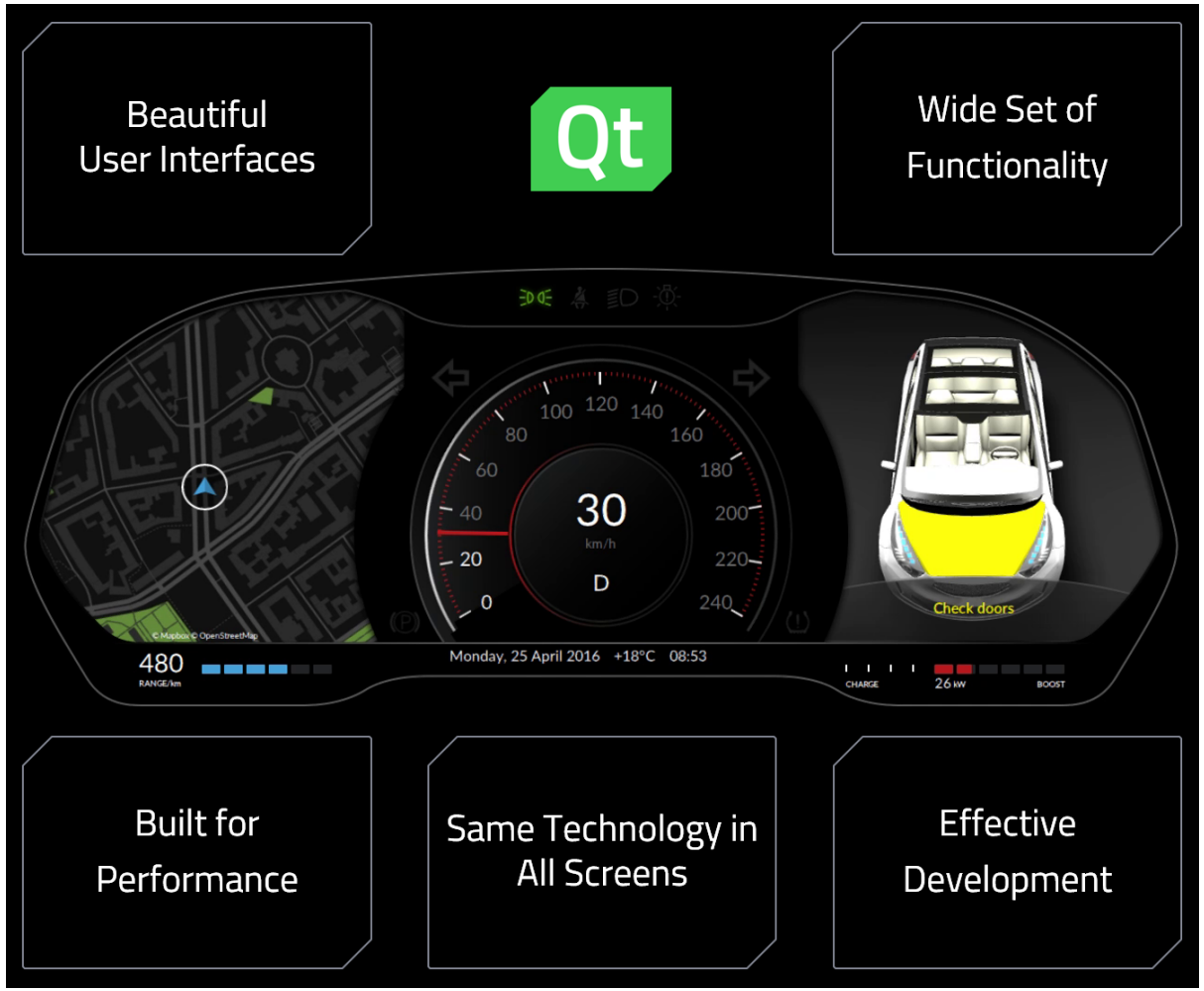
Targets**:

- Qt libraries < 10MB (Qt Core, Qt GUI, Qt Quick, Qt QML, Qt Network)
- RAM < 16 MB (depends heavily on the application)
- CPU < 300MHz ARM
- OpenGL and Raster

*** Initial research, subject to change*

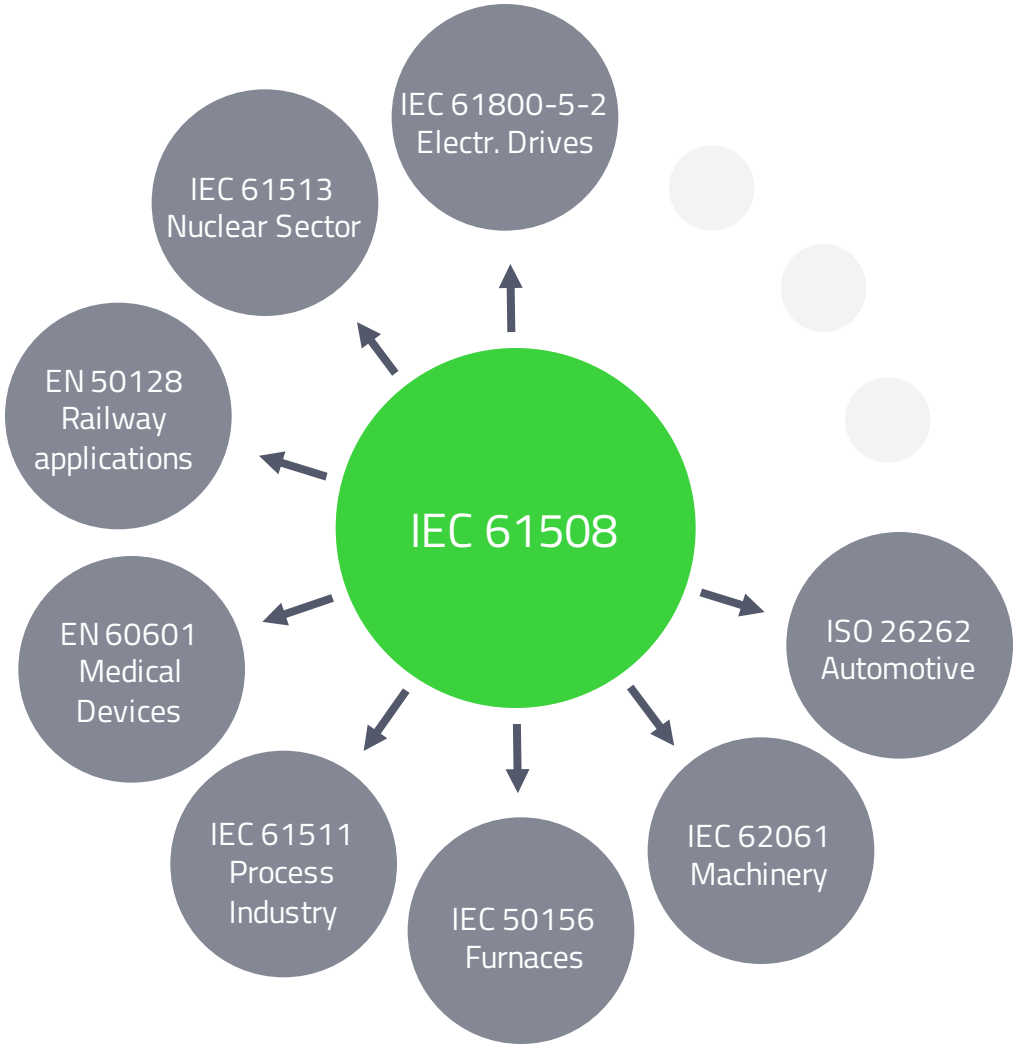
Cluster Research

- Researching ongoing for using Qt in digital instrument clusters
- First demo shown at Qt World Summit 2015, updated version at Embedded World 2016
  - i.MX6 CPU running embedded Linux
  - Fresh UI concept following current design trends
  - Testbed for new Qt features: Qt 3D, CanBus etc
- Leverage Qt Lite for resource optimization
- Boot time optimization
- Embedded Linux or RTOS used
- System level certification for functional safety (ISO 26262 ASIL B) requirements
- Main value proposition: Use Qt in all screen of the vehicle (i.e. migrate Qt from IVI to cluster as well)



Relevant Safety Standards for Qt Business

- Main standard of functional safety is IEC 61508
  - In essence all other standards for various industries are based on it
- Key industry standards
  - Automotive: ISO 26262
  - Medical Devices: IEC 60601
- Examples of other industry standards
  - Railway software: EN 50128
  - Avionics software (USA): DO-178B
  - Machine control: IEC 62061
  - Agricultural machines: ISO 25119
  - Nuclear: IEC 61513

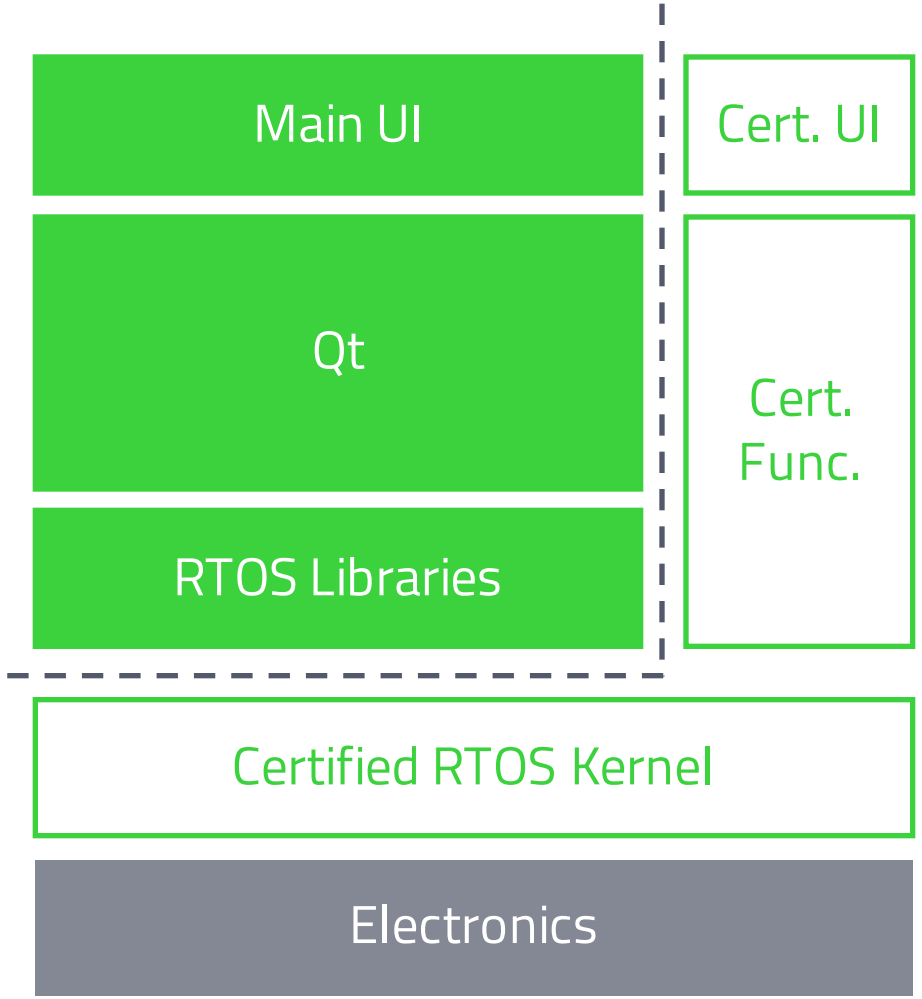


Creating a Certified System with Qt

- Safety critical functionality needs to be adequately separated, otherwise full software of the embedded systems must be certified
  - Level of separation is dependent upon the required SIL/ASIL level
- With separation, Qt can be used in a system requiring certification without certification of Qt libraries
- Architectures for separation of safety critical functionality
  - 1. Use a certified RTOS that can separate the certified and non-certified processes
  - 2. Use a certified Hypervisor to run two different operating systems, one for safety critical and one for other parts

Certified RTOS for Separation

- Using an RTOS that can separate safety critical and other processes
- Certification only for the safety critical parts
- Certified RTOS and toolchain saves time and effort in system level certification
- UI elements can be separated for example using HW layers or by the RTOS compositor
- In some designs, a certified UI may not be necessary at all, or can be arranged using a separate display / warning light



Hypervisor for Separation

- A Hypervisor can be used to run separate OS for certified and non-certified functionality
- Certified functionality can run on a much simpler RTOS that would be needed to run Qt
- Non-certified functionality can run for example on embedded Linux
- Operating systems can share resources and data
- Certified functionality can be assigned to a dedicated CPU core