

# Meet Qt 5.7

May 2016



# Contents

- › 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 and productiveness, 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

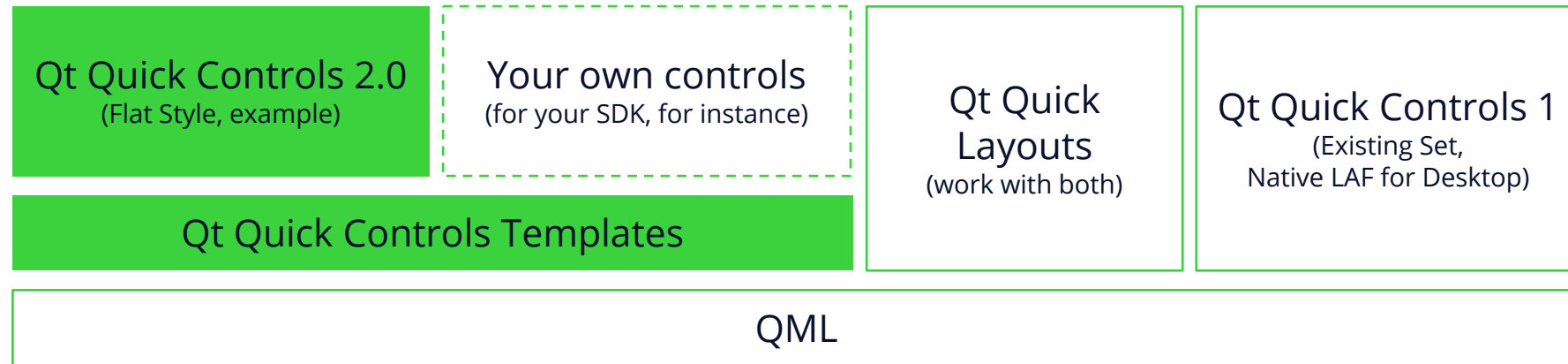
# Introducing Qt 3D – Fully Supported with Qt 5.7

- › 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



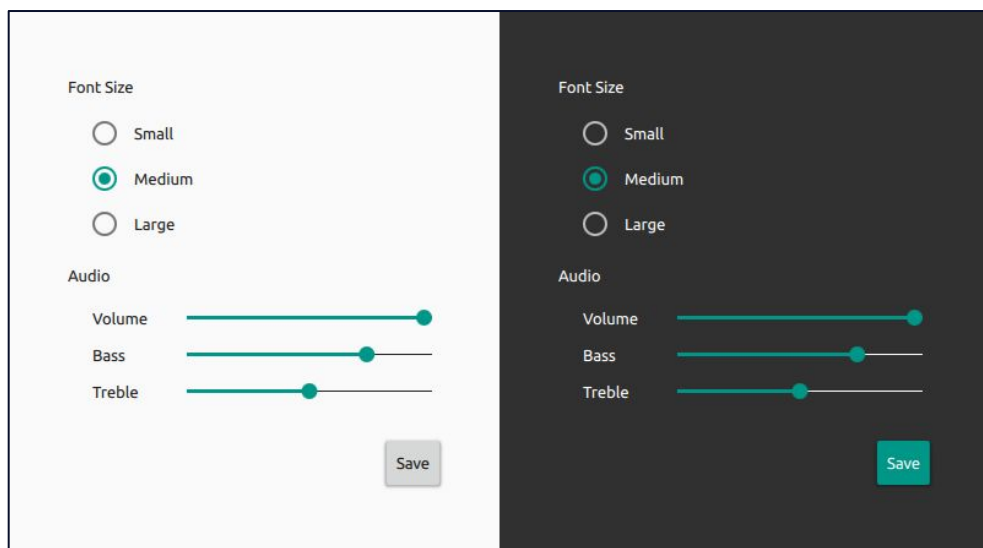
# Productivity Out-of-the-Box – Qt Quick Controls 2.0

- › 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)

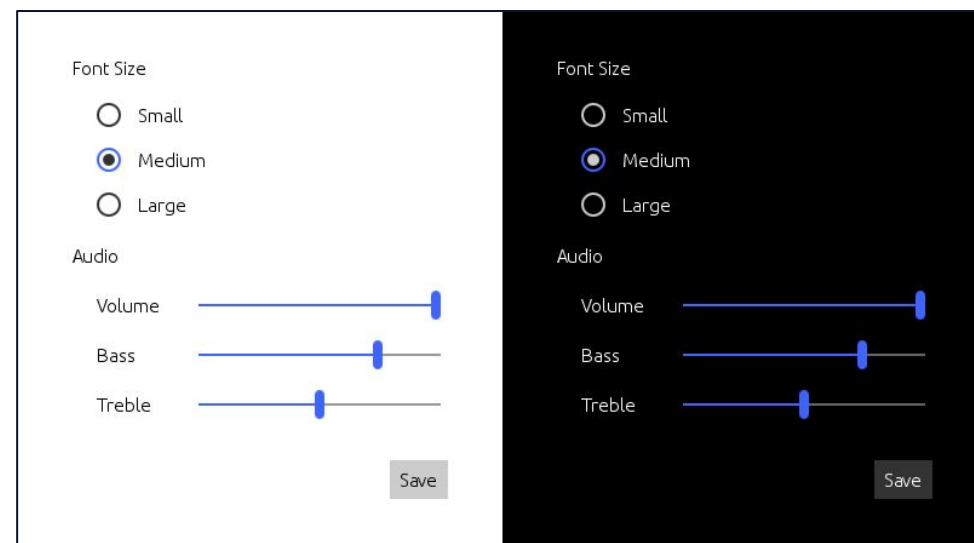




# Qt Quick Controls 2.0 - Examples



Google Material  
Design



Microsoft Universal  
Design



# The Framework for Modern C++

The background image shows a laptop screen with a software development environment. The interface includes a sidebar with a file explorer, a central dashboard with several circular gauges and charts, and a code editor area on the right. The entire image is overlaid with a semi-transparent green filter.

# 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

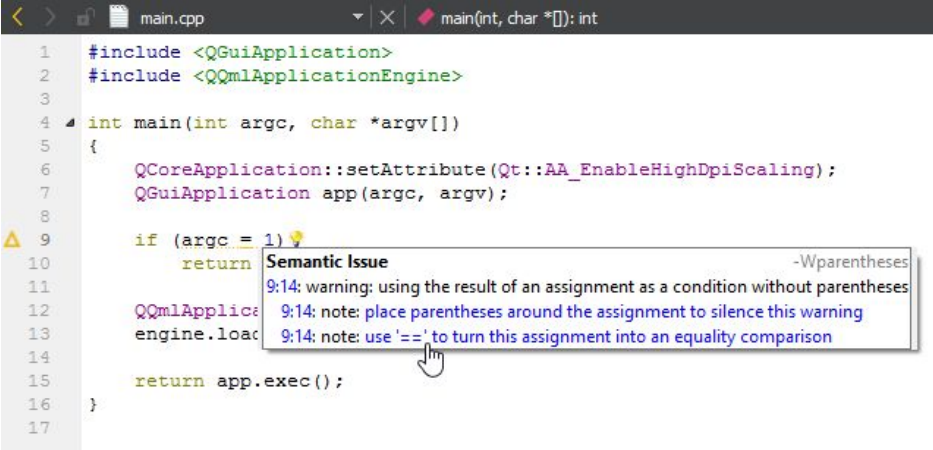


# Get Ahead of The Rest

- › 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



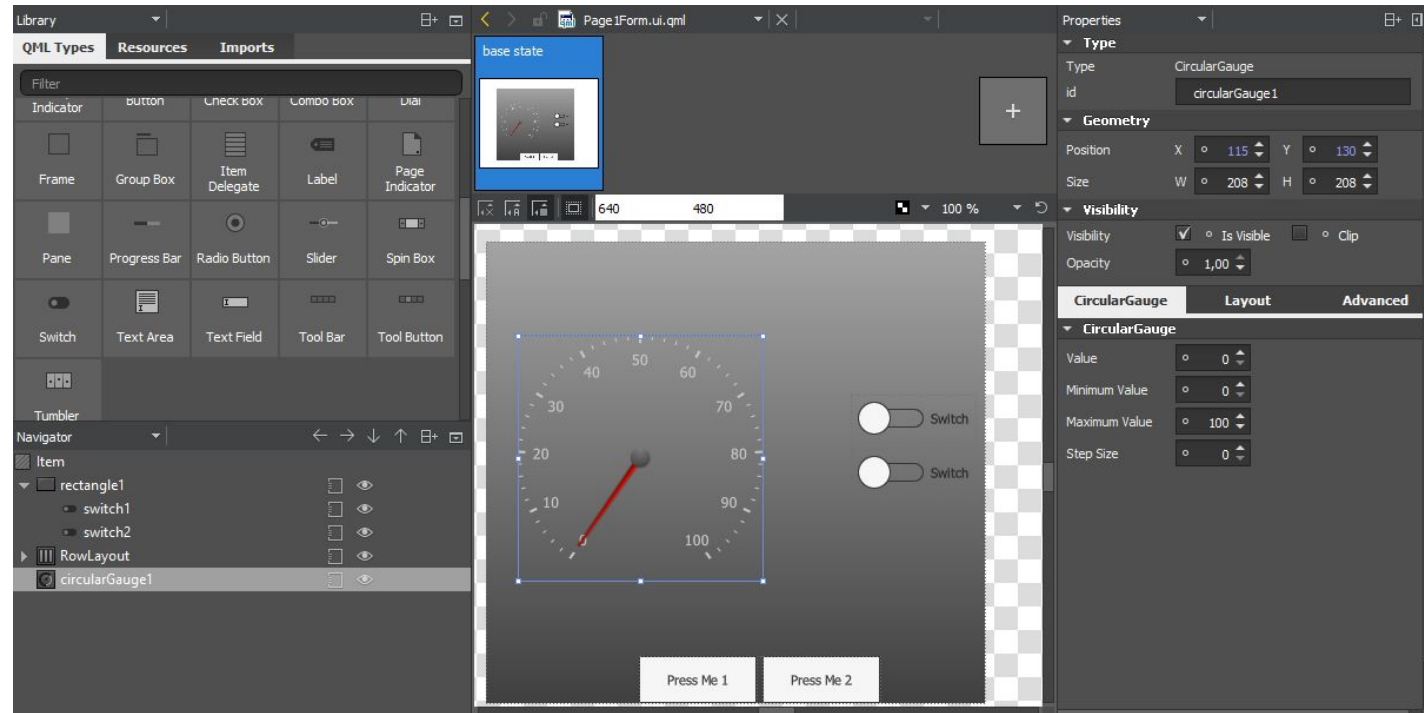
```
1 #include <QGuiApplication>
2 #include <QQmlApplicationEngine>
3
4 int main(int argc, char *argv[])
5 {
6     QCoreApplication::setAttribute(Qt::AA_EnableHighDpiScaling);
7     QGuiApplication app(argc, argv);
8
9     if (argc = 1)
10         return 0;
11
12     QQmlApplicationEngine engine;
13     engine.load(QUrl(QStringLiteral("qrc:/main.qml")));
14
15     return app.exec();
16 }
17
```

Semantic Issue -Wparentheses  
9:14: warning: using the result of an assignment as a condition without parentheses  
9:14: note: place parentheses around the assignment to silence this warning  
9:14: note: use '==' to turn this assignment into an equality comparison

*Find problems early with Qt Creator and CLang static analyzer*

# 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.*

# Qt Creator 4.0 – QML Profiler

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

The screenshot displays the Qt Creator 4.0 QML Profiler interface. The top-left pane shows the project structure for 'MyApplication', including files like 'main.qml', 'Page1.qml', and 'Page1Form.ui.qml'. The top-right pane shows the QML code for 'CircularGaugeStyle.qml', with the current line being: `image { source: "images/knob.png" anchors.centerIn: parent scale: {`. The main area is dominated by a flame graph visualization showing the execution time of various QML components. A call stack window is open, displaying the following details for the selected component:

Property	Value
Details:	QtQuick/Image
Type:	Create
Calls:	1
Total Time:	332.082 ms
Mean Time:	332.082 ms
In Percent:	26%
Location:	CircularGaugeStyle.qml:423

The flame graph shows a hierarchy of components, with 'QtQuick/Image (Create, 26%)' highlighted in red. Other components visible in the stack include 'QtQuick/Loader (Create, 26%)', 'QtQuick...', 'sourceComponent: \_\_style ? \_\_style.panel : null (Binding, 3...', 'property QObject \_\_style: styleLoader.item (Binding, 37%)', and 'QtQuick/Loader (Create, 37%)'. The bottom status bar indicates the elapsed time of 10.4 s and provides navigation options for the profiler.

# 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

The screenshot shows the Qt Creator interface with the 'Build & Run' tab selected. The 'Build Kits' section displays four kits: 'Boot2Qt 5.6.0 Emulator', 'Boot2Qt 5.6.0 Freescale SABRE SD i.MX6', 'Boot2Qt 5.6.0 Toradex Colibri iMX6', and 'Desktop Qt 5.6.0 MinGW 32bit'. Each kit has 'Build' and 'Run' buttons. Below the kits, the 'Build Settings' for the selected kit are shown, including 'Shadow build' and 'Build directory'. A 'Build Steps' section shows the command: `qmake.exe Qt5_CinematicExperience.pro -r -spec devices\linux-oe-generic-g++ "CONFIG+=debu`. Callout boxes provide descriptions for each kit: 'Qt build environment for emulator on Windows system', 'Qt build environment for Target Board A.', 'Qt build environment for Target Board B.', and 'Qt build environment for Windows OS.'. A purple callout box on the right says 'Add More' and 'You can add more as you need'.

# 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)
- › **New Licensing Offering**
  - › 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**
- › 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

# Overview of 2015-2016 Qt Releases

