Qt Compliance Test application

for embedded devices

Specification v1.0

Performance test set specification v1.0

Compliance test application (QTBUG-70286)

- **Ease of use**
    - Tester should be able to just open the project in Qt Creator, click build and run. The project should then build and deploy itself to the target hardware and run automatically.
        - Results should be shown in the application output window and be stored somewhere. Maybe on the target device, where tester then can copy the file to their computer for future inspection.
        - This all is assuming, that compilers, target device etc. have been setup properly.
    - Results should be simple Pass/Fail
- **Recognize or test hardware capabilities so that certain tests can be enabled and disabled based on hw availability.**
    - GPU detection
        - GPU capabilities (OpenGL(ES) version support, Vulkan support etc. detection)
    - Bluetooth detection
    - NFC support detection
    - WLAN detection
    - Sound Card detection
        - Detect that ALSA output device is created
    - Camera detection
        - Might need user input though
- **Error handling**
    - Test crashes etc. must be handled accordingly.
    - If a crash occurs, stack trace should be stored and test marked as a failure (or inconclusive; however, test crashing is most likely due to something not working right on the device)
- **Following must be tested:**
    - Most features refer to https://doc.qt.io/qt-5/qmltypes.html and https://doc.qt.io/qt-5/classes.html. Please refer these when implementing tests.
    - ***QML features***:
        - First and foremost, most of the QML features are visual, so the tests should make sure that everything is rendered properly.
        - QML types (see https://doc.qt.io/qt-5/qtqml-qmlmodule.html)
            - Basic types can be created and their methods etc. work without problems

- Qt Wayland Compositor
  - Display server using wayland protocol can be created
  - Wayland clients can be shown in the display server and they work without problems.
  - Other modules can be tested in a wayland client (i.e. run the test suite as a wayland client)
- Qt 3D
  - Basic scene creation
  - Basic primitive models can be loaded
  - Shaders can be created and used (in addition to requiring a GPU, shaders usually have minimum OpenGL(ES) versions defined. A good idea would be to have multiple shaders requiring different OpenGL(ES) versions.
  - It should be possible to display QML elements on top of the 3D scene.
- Qt Audio Engine
  - Positional sounds can be created and played back without problems
- Qt Bluetooth
  - Bluetooth devices can be searched
  - Requires Bluetooth
- Qt Charts
  - Charts of different types can be created
  - Charts are rendered properly
- Qt Data Visualization
  - Data visualizations can be created and displayed without problems
  - This requires GPU
- Qt Graphical Effect
  - Graphical effects can be applied
- Qt Location
  - A map can be created and displayed without errors. User location can be acquired, provided that an internet connection is available.
  - Different backend plugins (e.g. mapbox and mapbox_gl) work
    - Some backends might require GPU
- Qt Multimedia
  - Audio and video can be loaded
  - Audio is played without errors (if soundcard is detected and ALSA device is created)
  - Video is played without errors (this most likely requires GPU)
  - Post processing/shaders can be applied to a video
  - If camera is found, it can be accessed and used

- Qt NFC
  - NDEF messages can be acquired or sent
  - Requires NFC support
  - Due to requiring another device, this test might not be feasible enough to be done automatically
- Qt Positioning
  - Position data can be acquired (from Data file? This needs clarification as the implementation might not be feasible)
- Qt QML Models
  - Models can be created and used in appropriate containers
- Qt QML State Machine
  - State machine can be created from a SCXML (http://www.w3.org/TR/scxml/)
- Qt Quick Dialogs
  - Dialogs can be created
  - Dialogs show up properly
- Qt Quick Shapes
  - All shapes can be created
  - Gradients can be used
  - Shapes and gradients display properly
- Qt Quick Test
  - A unit test can be created and run without problems
- Qt Quick Controls
  - Modules, that should be tested: https://doc.qt.io/qt-5/qtquick-controls2-qmlmodule.html and https://doc.qt.io/qt-5/qtquick-controls-qmlmodule.html
  - Test that themes work
    - Some of the built-in themes (at least Material) require GPU
  - You can look at Qt Quick Controls – Gallery example for a reference.
- Qt Remote Objects
  - A replica type can be created and registered to QML (C++ class, see https://doc.qt.io/qt-5/qtremoteobjects-qmlmodule.html for a minimal implementation)
  - Node can be created for the replica
  - Type’s properties can be read and changed
- Qt Quick Window
  - Windows can be created and closed
  - Display information can be acquired using Screen
- Qt WebChannel
  - QML objects can be accessed from a HTML client
  - Properties, signals and slots can be used from the client

- Qt WebEngine
  - WebEngineView can be created
  - A webpage is loaded and displayed successfully, provided an internet connection is found.
- Qt WebSockets
  - A web socket can be created
  - A web socket server can be created
  - Web socket can connect to the server
  - Text messages can be sent and received
  - Binary messages can be sent and received
- Qt XML Patterns
  - XmlListModel can be created from XML file
  - Created model can be viewed using an appropriate container
- Qt Device Utilities module
  - QtDeviceUtilities.BluetoothSettings (If Bluetooth is supported on the device)
    - Bluetooth module can be turned on and off
    - Bluetooth devices are detected
    - Connection to a device via BT
  - QtDeviceUtilities.DisplaySettings (If an appropriate screen is connected)
    - Display brightness value can be read and set. This requires a compatible display, so implementation might be a bit difficult.
    - Physical Screen Size can read and set.
  - QtDeviceUtilities.LocalDeviceSettings
    - Device can be rebooted or powered off
    - This might need some tricks
  - QtDeviceUtilities.LocaleSettings
    - System Locale can be read and set
    - Locale specific details, such as date format, currency etc. change when locale is changed
  - QtDeviceUtilities.NetworkSettings
    - Connecton to a wired network is possible
    - WLAN (if supported on the device) can be turned on or off
    - When turned on, WLAN networks are found
  - QtDeviceUtilities.TimeDateSettings
    - NTP (Network Time Protocol) can be turned on or off
    - Date and time can be read (and set?)
    - Timezone can be read and set
  - QtDeviceUtilities.SettingsUI
    - This is just a reference implementation of all the different Qt Device Utilities modules.

- Qt Virtual Keyboard and different input box types
  - Writing to different types of input fields (text, numeric, password etc.)
  - Changing writing language and ensuring that the input is done correctly (e.g. changing to Japanese and making sure that input method can chain kanas together to form a kanji)
  - All languages are rendered correctly (e.g. changing to Japanese and making sure that appropriate characters are displayed)
- Other
  - Labs modules are not included due to them being mostly technological previews
  - Sensors module is not included because additional hardware is needed
  - Qt NFC can not be tested automatically due to it requiring another NFC device
  - Qt Gamepad is left out due to tests requiring a gamepad
  - Qt Purchasing is not included since it is supported only in Android, iOS and macOS

- ***C++ features***
    - Most of the module specifications are shared with corresponding QML ones. Feel free to even use the two together when testing, however, make sure that both QML and C++ functionalities are tested properly.
    - Qt 3D
        - Basic scene creation
        - Basic primitive models can be loaded
        - Shaders can be created and used (in addition to requiring a GPU, shaders usually have minimum OpenGL(ES) versions defined. A good idea would be to have multiple shaders requiring different OpenGL(ES) versions.
    - Qt Bluetooth
        - Bluetooth devices can be discovered
        - Bluetooth server can be created
        - Requires Bluetooth
    - Qt Charts
        - Charts of different types can be created
        - Charts are rendered properly in Graphics View Framework (https://doc.qt.io/qt-5/graphicsview.html)
    - Qt Concurrent
        - Multi-threading programs can be created using QtConcurrent
        - Programs run without any problems
    - Qt Core
        - All classes can be used without errors. See https://doc.qt.io/qt-5/qtcore-module.html for a full list of classes inside Qt Core.
    - Qt D-Bus
        - Applications can use the D-Bus protocol to communicate with each other
    - Qt Data Visualization
        - Data visualizations can be created and displayed without problems
        - This requires GPU
    - Qt GUI
        - Most classes can be used without errors. See https://doc.qt.io/qt-5/qtgui-module.html#details.
        - OpenGL and Vulkan can be used only if the target has a GPU.
        - Input event classes can be left out.
    - Qt Help
        - A simple help can be created
        - Help contents can be searched
        - Contents are rendered properly

- Qt Location
  - A map can be created and displayed without errors. User location can be acquired, provided that an internet connection is available.
  - Different backend plugins (e.g. mapbox and mapbox_gl) work
    - Some backends might require GPU
- Qt Multimedia
  - Audio and video can be loaded
  - Audio is played without errors (if soundcard is detected and ALSA device is created)
  - Video is played without errors (this most likely requires GPU)
  - Post processing/shaders can be applied to a video
  - If camera is found, it can be accessed and used
- Qt NFC
  - NDEF messages can be acquired or sent
  - Requires NFC support
  - Due to requiring another device, this test might not be feasible enough to be done automatically
- Qt Network Authorization
  - Authentication with OAuth works.
  - This might not be feasible to implement due to requiring a server, which accepts OAuth requests.
- Qt Network
  - Local, TCP, SCTP and UDP servers and sockets can be created
  - Messages can be transported between all protocol servers and sockets.
  - QHostInfo can be used to get host name lookup results
- Qt Platform Headers
  - QEglFSFunctions can access platform-specific functionality of the eglfs platform plugin
  - QLinuxFbFunctions can access platform-specific functionality of the linuxfb platform plugin
  - QEGLNativeContext can access EGL context and display handle
- Qt Positioning
  - Position data can be acquired (from Data file? This needs clarification as the implementation might not be feasible)
- Qt QML
  - Javascript can be evaluated using QJSEngine
  - QML can be instantiated using different ways presented in this module
  - QML is rendered properly
- Qt Quick
  - QQuickWindow and QQuickView can be instantiated and shown
  - Qt Quick can be embedded to C++ applications

- Qt Quick Controls
  - Application QML style can be changed from C++ code
- Qt Quick Test
  - A unit test can be created and run without problems
- Qt Quick Widgets
  - QQuickWidget can be created
  - QML can be loaded to the widget
  - Widget can be used with other widgets
- Qt Remote Objects
  - A replica type can be created
  - A Host node can be created
  - Node can be created for the replica
  - Type’s properties can be read and changed
- Qt SCXML
  - State machine can be created from a SCXML (http://www.w3.org/TR/scxml/)
- Qt SQL
  - SQL Database can be accessed
  - Entries in database can be added, removed and altered
- Qt SVG
  - SVG images can be loaded and displayed
- Qt Serial Port
  - Information from serial ports (if any) can be retrieved
- Qt Test
  - Unit tests can be created and run
- Qt UI Tools
  - UI file can be loaded dynamically to the application
  - Loaded UI can be displayed
- Qt Wayland Compositor
  - Display server using wayland protocol can be created
  - Wayland clients can be shown in the display server and they work without problems.
  - Other classes can be tested in a wayland client (i.e. run the test suite as a wayland client)
- Qt WebChannel
  - QObjects can be accessed from a HTML client
  - Properties, signals and slots can be used from the client
- Qt WebEngine
  - WebEngineProfile can be created in C++ and applied to Widget and QML WebViews
  - QWebEngineView can be created and used with other widgets
  - A webpage is loaded and displayed successfully, provided an internet connection is found.
  - QtWebView can be used to intialize QML WebView

- Qt WebSockets
    - A web socket can be created
    - A web socket server can be created
    - Web socket can connect to the server
    - Text messages can be sent and received
    - Binary messages can be sent and received
- Qt Widgets
    - Widgets can be created (list of all widgets: https://doc.qt.io/qt-5/qtwidgets-module.html)
    - Widgets are rendered correctly
- Qt XML
    - XML file can be laoded and parsed
- Qt XML Patterns
    - XmlNodeModel can be created from XML file
    - XML Schemas can be validated
- Qt Device Utilities
    - The following classes should at least test the features specified.
        - Qt Bluetooth Settings C++ Classes (If bluetooth is supported on the device)
            - Bluetooth module can be turned on and off
            - Bluetooth devices are detected and maybe
            - Connection to a device via BT
        - Qt Display Settings C++ Classes (If an appropriate screen is connected)
            - Display brightness value can be read and set. This requires a compatible display, so implementation might be a bit difficult.
            - Physical Screen Size can read and set.
        - QtLocalDeviceSettings C++ Classes
            - Device can be rebooted or powered off
            - This might need some tricks
        - Qt Locale Settings C++ Classes
            - System Locale can be read and set
            - Locale specific details, such as date format, currency etc. change when locale is changed
        - Qt Network Settings C++ Classes
            - Connecton to a wired network is possible
            - WLAN (if supported on the device) can be turned on or off
            - When turned on, WLAN networks are found
        - Qt Time and Date Settings C++ Classes
            - NTP (Network Time Protocol) can be turned on or off
            - Date and time can be read (and set?)
            - Timezone can be read and set

- Other
    - Qt Designer classes are left out due to them being desktop specific
    - Qt Gamepad classes are left out due to tests requiring a gamepad
    - Qt Mac Extras classes are left out due to them requiring a macOS
    - Qt OpenGL classes are left out due to them being mostly deprecated
    - Qt Print Support classes are left out due to printer requirement
    - Qt Script and Qt Script Tools are left out because they are not actively developed and provided only for backwards compatibility with Qt 4
    - Qt Sensors classes are not included because additional hardware is needed
    - Qt Serial Bus is not included due to needing additional hardware for testing
    - Qt Virtual Keyboard C++ classes are mostly meant for input method development and thus won’t need testing in a device
    - Qt Windows extras is not included due to Windows needed for testing the functionalities
    - Qt Purchasing is not included since it is supported only in Android, iOS and macOS

Performance test set (QTBUG-70285)

- Ease of use
  - Tester should be able to just open the project in Qt Creator, click build and run. The project should then build and deploy itself to the target hardware and run automatically.
    - This all is assuming, that compilers, target device etc. have been setup properly.
- Separate test application from the compliance test application. However, compliance application should be able to launch performance test set.
- This test set is rather large and complex to do in a couple of months' time. Therefore, a subset of most critical aspects should be done first and a more comprehensive suite can be done later.
  - The first version of the performance test set should at least include *qmlbench*, an application that tests qml features for performance regressions. See https://codereview.qt-project.org/admin/repos/qt-labs/qmlbench
  - At least one 3D test set should be also done in order to test 3D performance of devices which have a GPU.
- If compliance test suite runs the perf test set, results should be in a file separate from the compliance test suite. One possibility would be to also have a separate program for thorough analysis of the performance results.
- The performance test application should be done in a way that makes it easy to expand in the future.
- Qt-benchmark (https://git.qt.io/kahormi/qt-benchmark) can be used as a basis for the test application. The bug report includes a master thesis draft (for full thesis, see http://jultika.oulu.fi/Record/nbnfioulu-201710112978), which was used as a basis for qt-benchmark.