KDAB

Qt DESKTOP DAYS

What’s new in Qt 6 on the desktop?

Qt Desktop Days 2020

Giuseppe D’Angelo

giuseppe.dangelo@kdab.com

The Qt, OpenGL and C++ experts

About me

- Senior Software Engineer, KDAB
- Developer & Trainer
- Qt Approver
- Ask me about QtCore, QtGui, QtQuick, …
  - And also about Modern C++, 3D graphics

The Road to Qt 6

KDAB

The Qt, OpenGL and C++ experts

Why Qt 6?

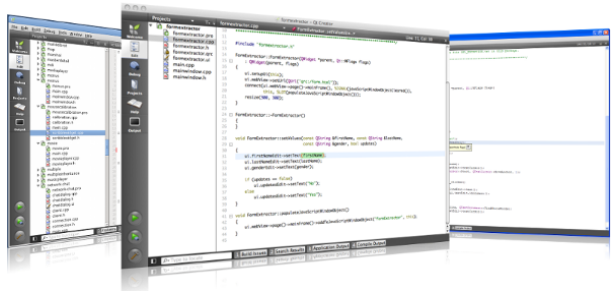
- Do architectural changes that simply cannot be done in Qt 5
- Binary compatibility break
  - Applications must be recompiled
- Re-engineer features
- But also do some necessary housecleaning, drop ballast

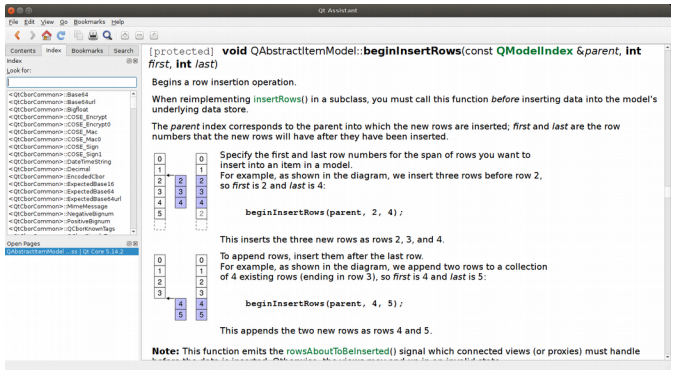
Design Goals

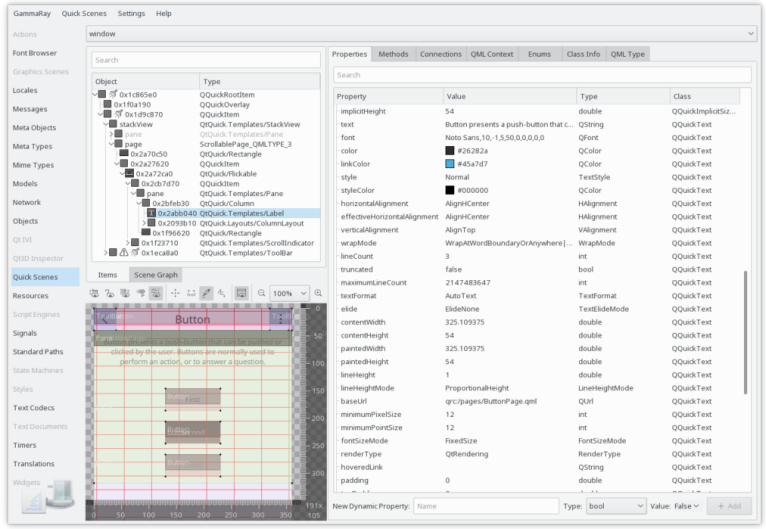
- **Keep as much (source) compatibility with Qt 5 as possible**
- Add property bindings in C++
- Improve QML & language bindings
  - Reduce overhead, increase type safety, compile to C++
- Tackle the changing landscape in 3D APIs
- Modularize Qt even more

Keep the Good Parts!

- Easy to use APIs
- General purpose, cross platform application framework
- Make 90% easy to achieve, and 99.9% possible
- Excellent developer support, documentation, tooling
- Nurture the ecosystem around Qt

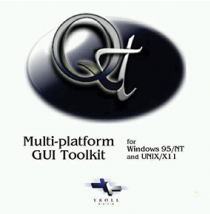






Looking ahead

- Qt 4: released 2005, EOL 2015
  - ~30 modules
- Qt 5: released 2012, EOL 2023
  - ~50 modules
- Qt 6: released 2020, EOL 20**??**
- How to plan for the next decade?



Technical foundations

- C++17
  - MSVC 2019, GCC 8, Apple Clang
- CMake buildsystem for Qt
  - qmake still supported for end user applications
- 3D API abstraction (Qt RHI)

Release Plan

September 2020	October 2020	November 2020	December 2020
Alpha	Beta	Release Candidate	**Qt 6.0 Final Release**

- Qt 6.0 feature freeze reached
- Binary weekly snapshots (already) available via the installer
- Reduced set of modules available for 6.0 release
  - More coming back in 6.1/6.2, or via the Marketplace

Qt Core

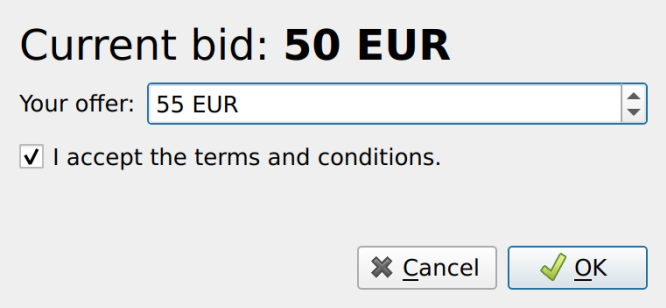
Property bindings in C++

- The defining feature of QML, available in C++
- Make properties *depend* on other properties without manually writing slots and setting up connections

Mandatory note: actual syntax / feature set still evolving

Property bindings in C++

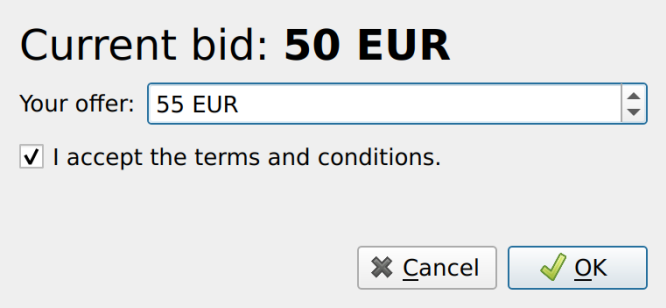
- The defining feature of QML, available in C++
- Make properties *depend* on other properties without manually writing slots and setting up connections



Mandatory note: actual syntax / feature set still evolving

Property bindings in C++

- The defining feature of QML, available in C++
- Make properties *depend* on other properties without manually writing slots and setting up connections




```
okButton->enabled = Qt::makePropertyBinding(

  [&](){ return offerEdit->value > currentOffer
                 && acceptBox->isChecked; }

);
```

Mandatory note: actual syntax / feature set still evolving

Qt container refactorings

- Support more than 2G elements
  - Index type is qsizetype
- Cleanups:
  - QList == QVector
  - QMap / QHash are single valued
    - QMultiMap / QMultiHash no longer inherit from QMap / QHash
  - New QHash, QMap, QPair, QVector implementations
  - QLinkedList dropped

String processing & I18N

- More flexibility in Qt string / byte array APIs
  - QStringView , QByteArrayView
  - QAnyStringView: views over UTF-8 / UTF-16 data
  - QStringTokenizer: non allocating string splitter
  - QStringRef / QRegExp superseded by QStringView / QRegularExpression
- Source code is UTF-8
- Support for Unicode 12, CLDR 36
- QtCore classes can now only deal with Unicode encodings
  - For other encodings: QTextCodec in Qt5Compat

Qt Widgets

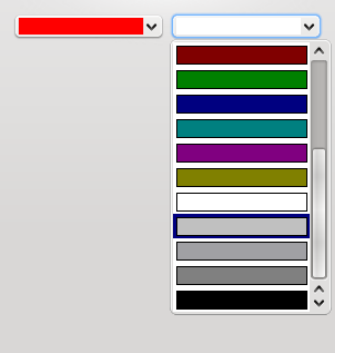
Widgets

- No new widget class has been added to Qt since 4.x!

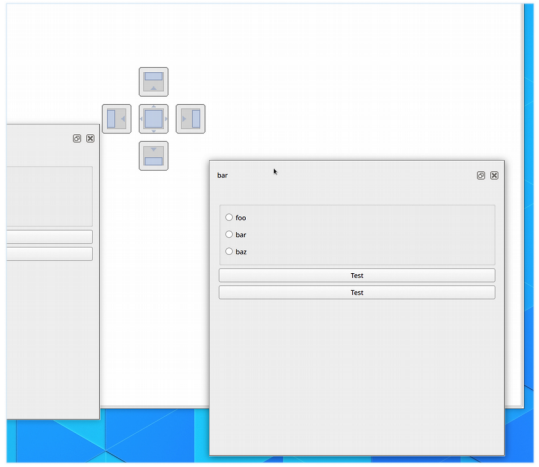
  What's going on?

Widgets: focus on stability

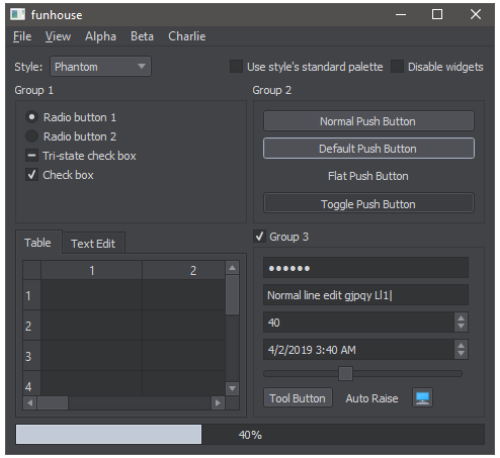
- Widgets are essentially frozen
  - Focus on long term stability, bugfixing, small API improvements
- Entry barrier for new features (esp. new widgets!) extremely high
- Qt Marketplace, KDE Frameworks, etc. offer many extra goodies
- Contributions welcome!



KColorCombo



KDDockWidgets



PhantomStyle

Still: keep up with new platform trends

- High-DPI in hybrid screen configurations must ”just work”
  - Incl. fractional factors
  - May require usage of non-native styles
- New native styles, easier theming
  - E.g. dark mode / themes support on all OSes
- Handle hybrid input systems
  - Mouse + Stylus + Touch + Voice + Virtual Keyboard + CIM + Eye + …
- Accelerate widget rendering & compositing through RHI
  - And/or platform native APIs

QML and Qt Quick

QML: new features in Qt 5.15

- `required` properties
- Inline components
- Registration of classes and objects from C++ to QML at build time
  - Enables checks on usage points
  - Improves tooling
  - qmake only in Qt 5.15, CMake support in Qt 6

```
class RadioController : public QObject
{
    Q_OBJECT
    QML_ELEMENT
    // ...
};
```

```
CONFIG += qmltypes
QML_IMPORT_NAME = com.kdab.carstuff
QML_IMPORT_MAJOR_VERSION = 1
```

Towards a better QML

- QML 2 has had a great run
- Still lots of room for improvement for big-scale (desktop) applications
- Lessons learned from QML 2:
  - Precompile as much as possible
  - JavaScript is fun, until it isn't
  - More build system / tooling support
  - Do more static analysis

Coming soon (6.1/6.2): QML 3

- JavaScript engine optional
- More static typing
- QML compiled to C++ code
  - By exploiting the new property system, new meta object, etc.

```
import QtQuick 2

Item {
    property bool active: checkBox.checked
}
```

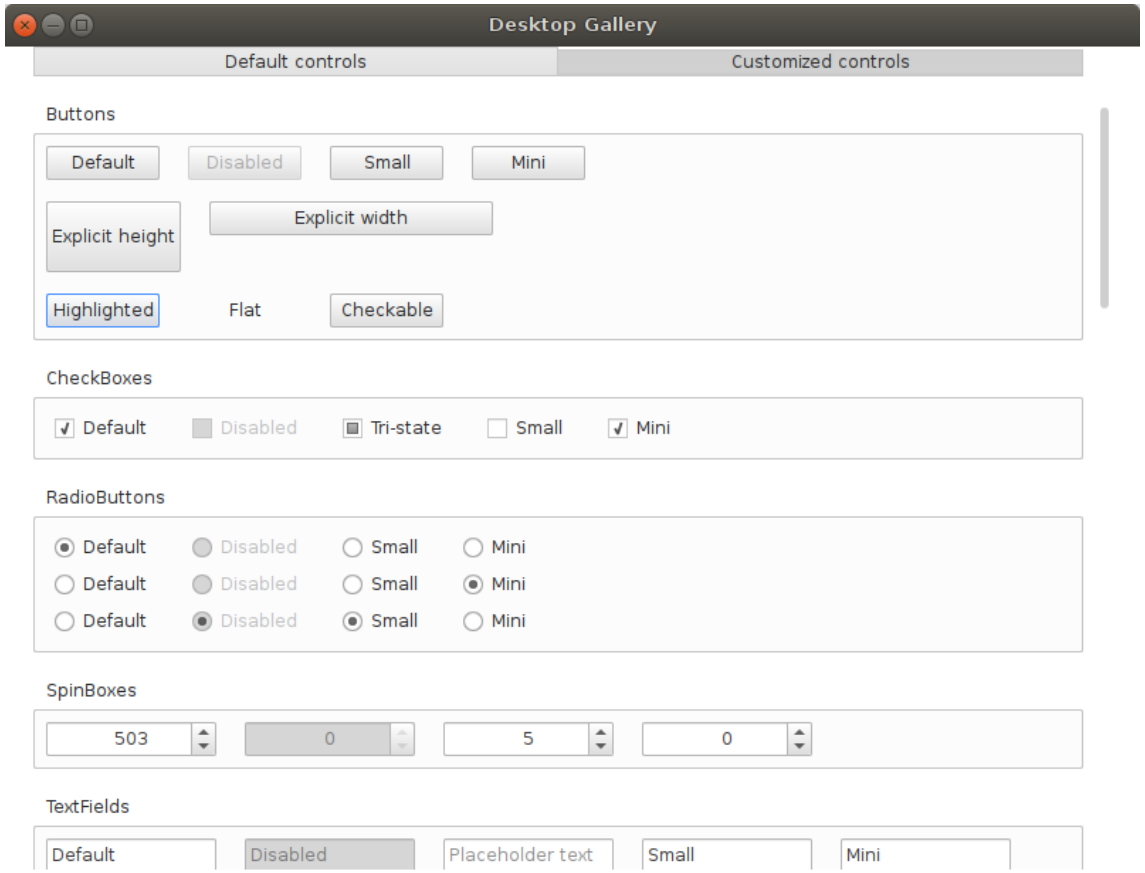
```
class MyItem : public QQuickItem
{
public:
    QProperty<bool> active;
};
```

Qt Quick

- API of elements is mostly unchanged
- Unified 2D/3D scenegraph
  - Tighter integration for 3D content
- Rendering happens exclusively through RHI
  - Targeting OpenGL (ES), Direct3D, Vulkan, Metal
  - Software rendering still available
  - Desktop applications mixing raw 3D and Qt Quick possibly affected
- Planned in Qt 6.x: C++ API for Qt Quick elements

Qt Quick Controls 2: Native Desktop Style

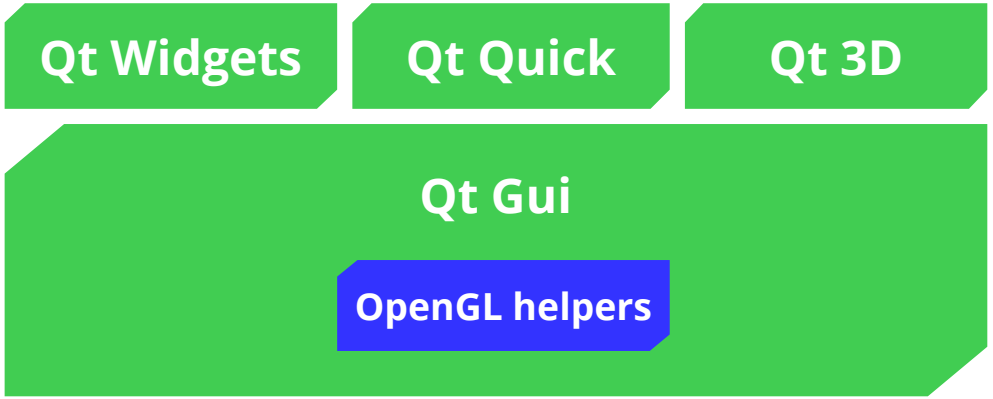
- Not every Qt Quick application wants a custom look and feel
- In Qt 6.0: native desktop style for Qt Quick Controls 2



3D Graphics

Qt and 3D graphics: a bit of history

- OpenGL, OpenGL, OpenGL
  - As a cross-platform toolkit, Qt has always had deep OpenGL integration
  - OpenGL widgets, QOpenGLContext, OpenGL helper classes, etc.
- Qt 5 bet on OpenGL (ES) as the universal enabler API for 3D
  - OpenGL first-class citizen in Qt APIs
  - QtGui featuring OpenGL classes
  - QtWidgets for OpenGL content
  - Qt Quick uses OpenGL for rendering

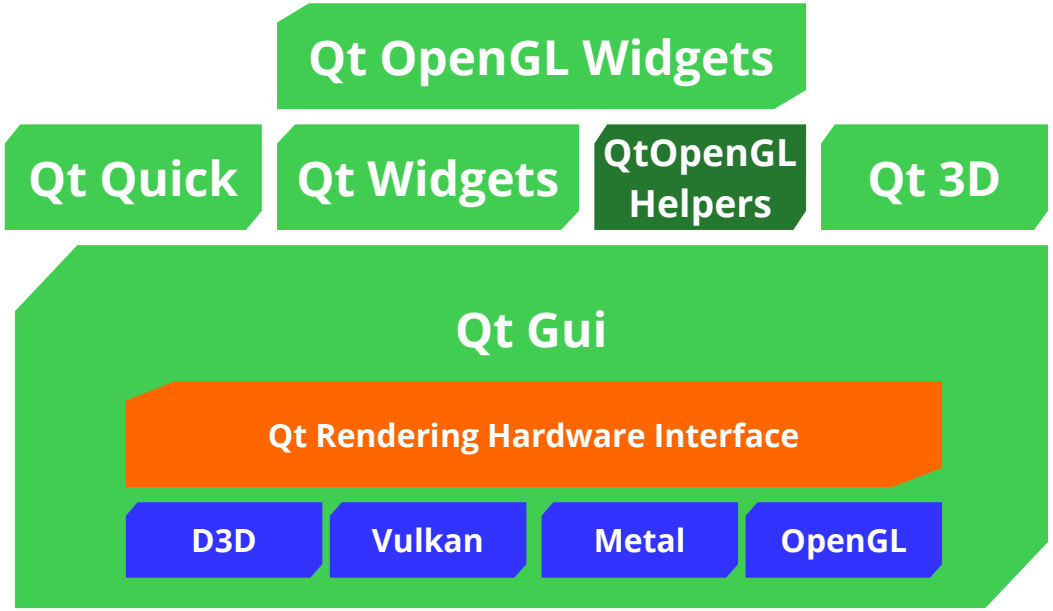


3D graphics today

- The OpenGL bet didn't quite pay off
- Today's world: multiple competing 3D standards
  - OpenGL
  - Vulkan
  - Metal
  - Direct3D
- Some support for Vulkan and Metal already available in Qt 5

Qt 6 strategy for 3D graphics

- Move everything but the core 3D classes out of QtGui, into their own libraries
  - Lots of OpenGL goodies still available
- Introduce RHI as foundational 3D API for Qt itself
  - Qt Quick, Qt 3D, etc. using it in 6.0



3D: summary

- Qt RHI initially (mostly) private API, for Qt's own libraries
  - Qt Quick, Qt Quick 3D, Qt 3D, etc.
- OpenGL enablers and higher-level classes are still available to applications
  - Some simply got moved; minor changes required
- New: low-level enablers for other 3D APIs, available in Qt Gui
- Application mixing raw 3D with Qt-rendered 3D may need some changes

How to upgrade to Qt 6?

The way towards Qt 6

- Recognize the success of Qt 5
  - Qt 5.15 has *Long Term Support* (3 years)
- Minimize disruptions for end-users towards Qt 6
- Not every Qt 5 module will be ready for 6.0
  - Notable desktop-related modules missing: QtWebEngine, QtVirtualKeyboard, QtMultimedia
  - Some of those may move to the Marketplace

Planning the upgrade

- Upgrade your software to Qt 5.15 LTS
- Upgrade your toolchain
- Enable the deprecation warnings in Qt, fix them all
- Use docs, changelogs, etc. to identify pain areas (“Important Behavior Changes”)
  - Have a plan to address them before getting your hands dirty

Migrate, not upgrade

- Upgrading towards Qt 6 is still a migration
  - Therefore: avoid adding features, doing refactorings, etc. while porting
  - If needed, port to Qt 5.15 features *before* porting to Qt 6
- In theory, have enough `#ifdefs` and/or build system support to keep your code work on both 5.15 and 6.x
- Some unsupported features moved to Qt5Compat module
  - Use it (temporarily!) to ease the transition

KDAB

Thank you!

Questions?