News from the Graphics Stack:
Improvements to the core OpenGL enablers in Qt 5.3 & 5.4

Laszlo Agocs
The Qt Company

Who am I?

- Senior software engineer at The Qt Company (formerly Digia) in Oslo, Norway
- OpenGL, graphics, windowing systems, platform integration
- Previously at ARM and Nokia

Why are we here?

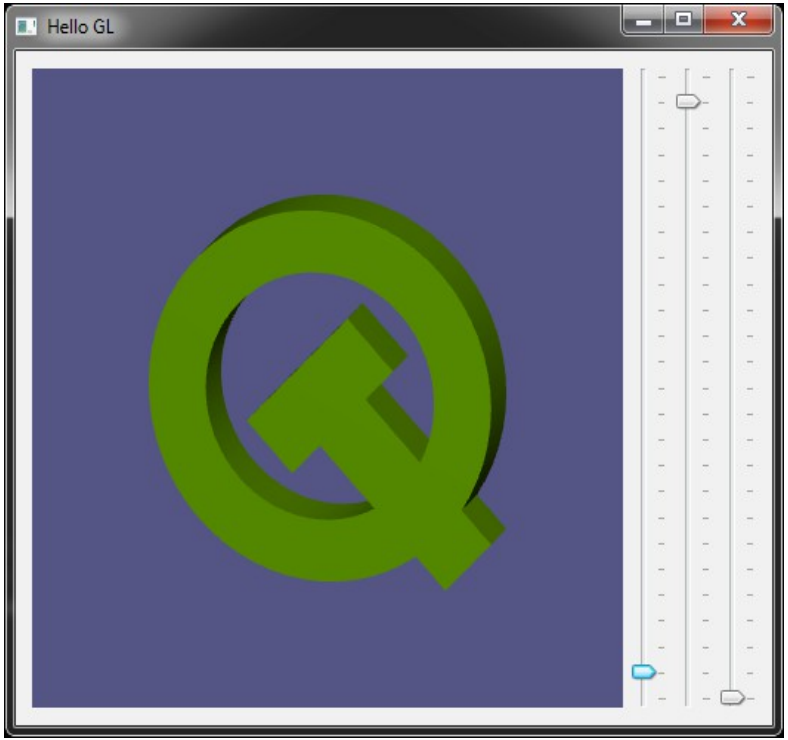
- QQuickWidget
- QQuickRenderControl
- QOpenGLWidget
- QOpenGLWindow
- QRasterWindow

The long road to QQuickWidget

OpenGL in Qt 4

```
class Widget : public QGLWidget, protected QGLFunctions
{
    void initializeGL() { initializeGLFunctions(); ... }
    void resizeGL(int w, int h) { ... }
    void paintGL() {
        // Render directly via GL or open a QPainter on 'this'
    }
};
```

Qt 4

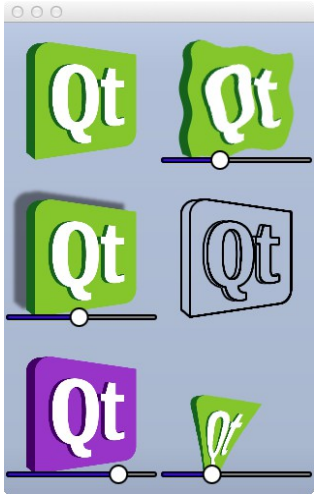


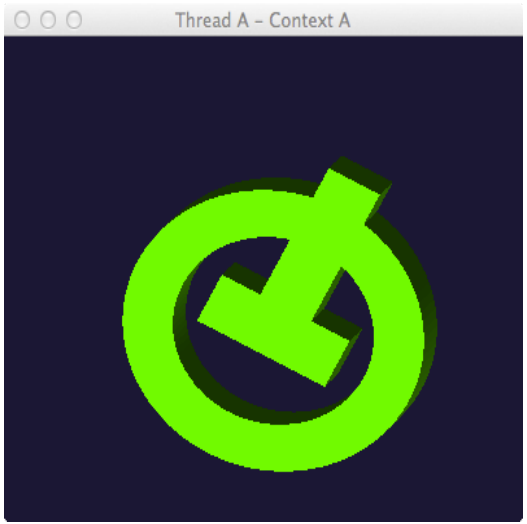
Native windows

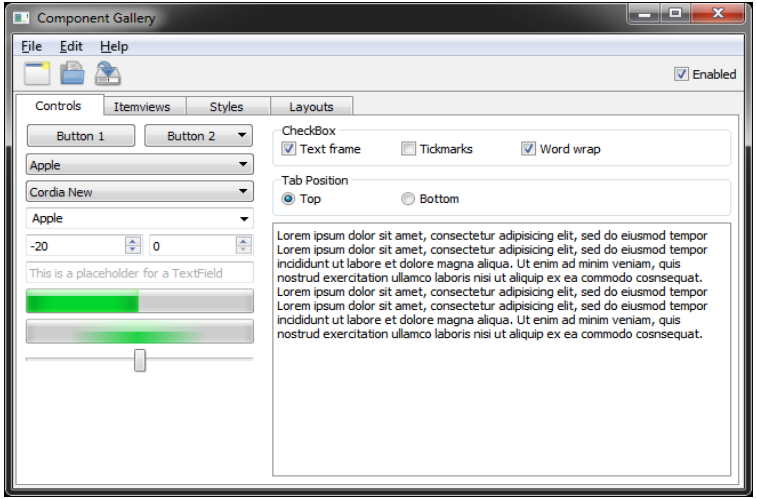
- Thread 00002130 HELLOGL
  - Window 002506B2 "Hello GL" QWidget
  - Window 00560460 "" QWidget
  - Window 001F085E "" QWidget
  - Window 00140802 "" QWidget
  - Window 0013088C "" QGLWidget
  - Window 001C054A "hellogl" QWidget
  - Window 00F103EE "hellogl" QWidget
  - Window 002306E4 "" QEventDispatcherWin32_Internal_Widget2007440147
  - Window 00120860 "MSCTFIME UI" MSCTFIME UI
  - Window 0006094E "Default IME" IME

Qt 5.0

- QWindow, QOpenGLContext, QSurfaceFormat
- QQuickWindow, QQuickView
- QGL*

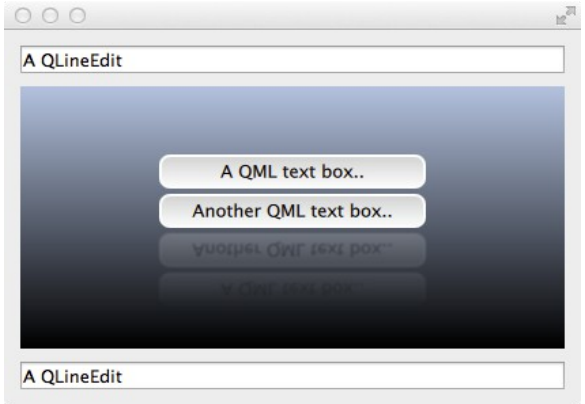


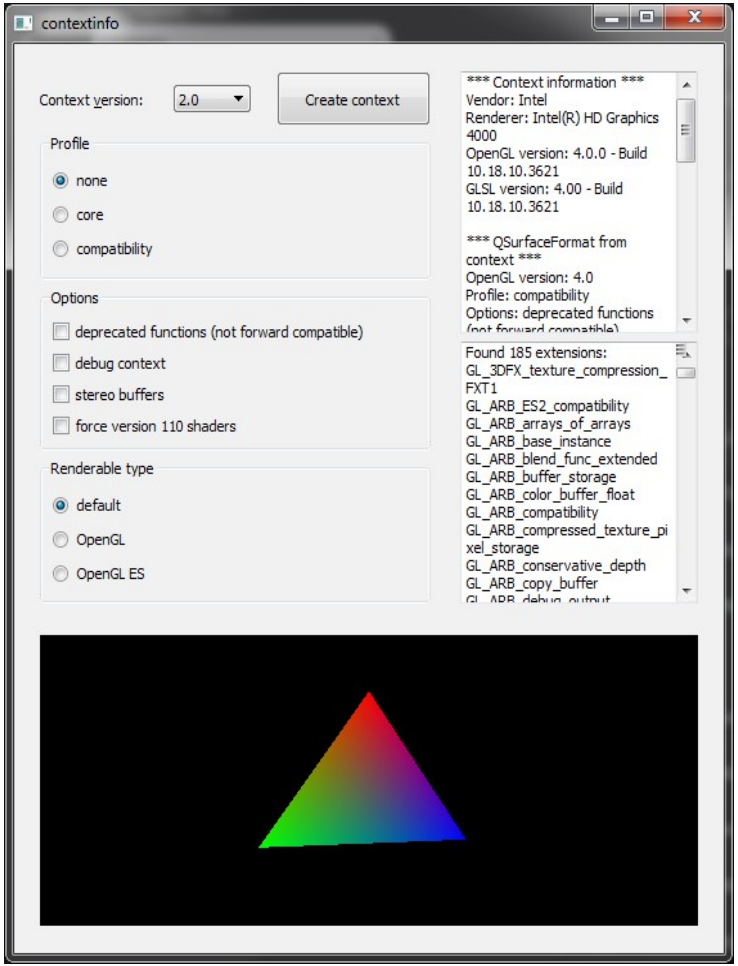




Qt 5.1

- QWidget::createWindowContainer()

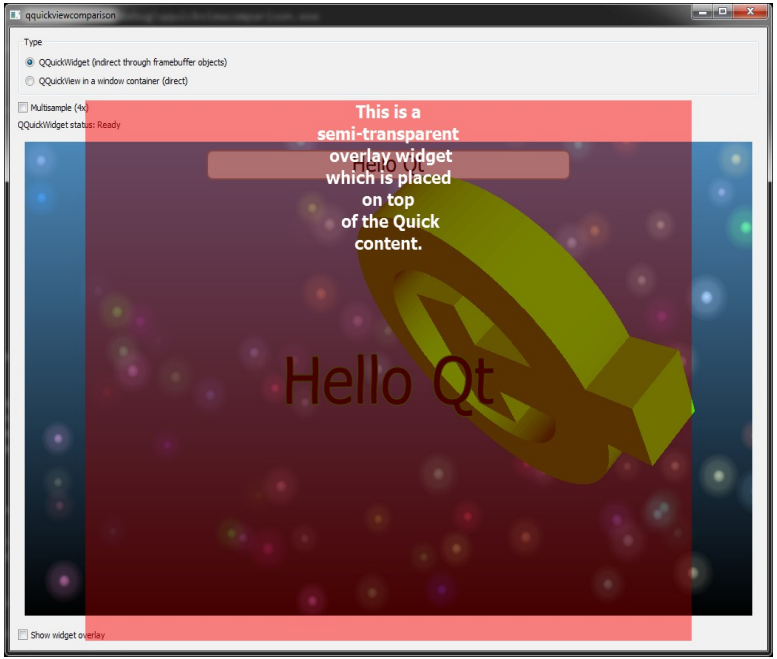




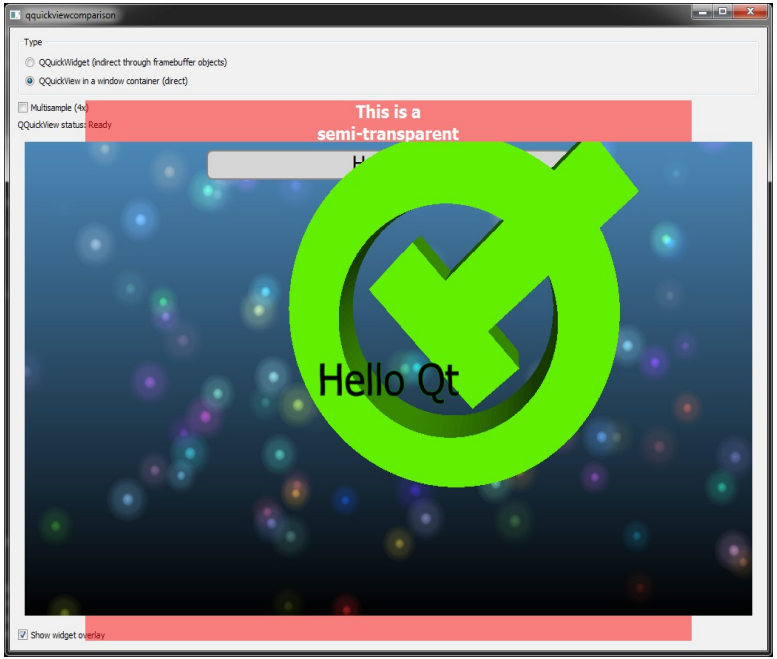
Qt 5.3

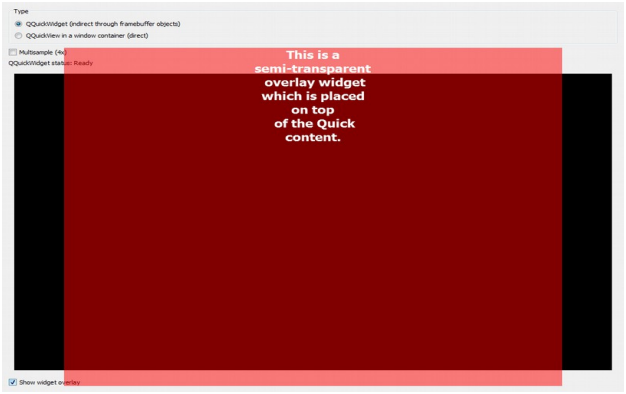
- New concept: Avoid native windows, render offscreen and composite in the widget stack
- Essential for embedded
- QQuickWidget

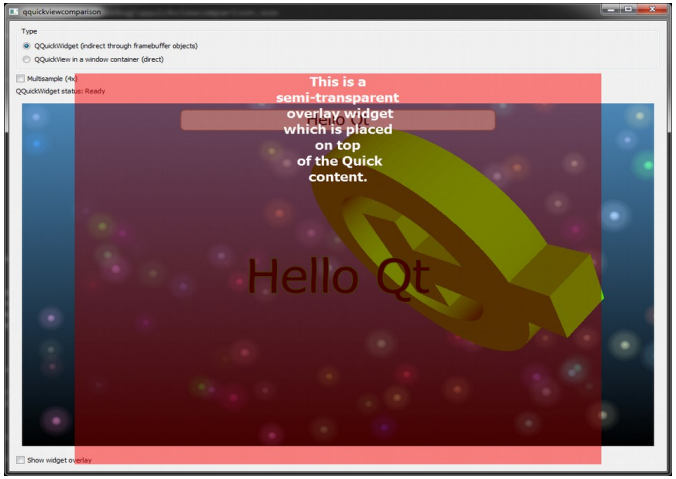
QQuickWidget



is always better than



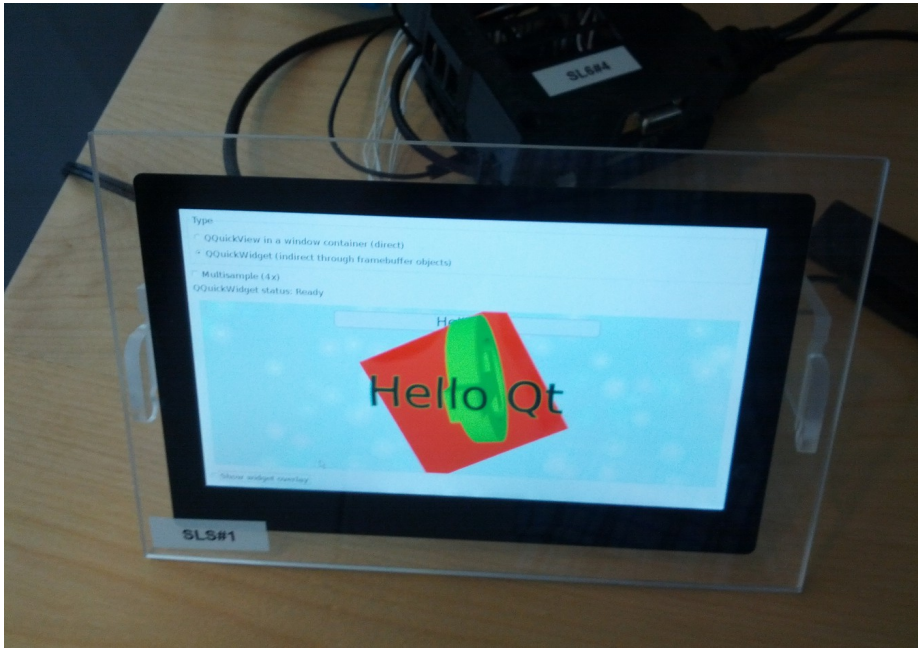




black has
alpha == 0

And it just works. Even on embedded.

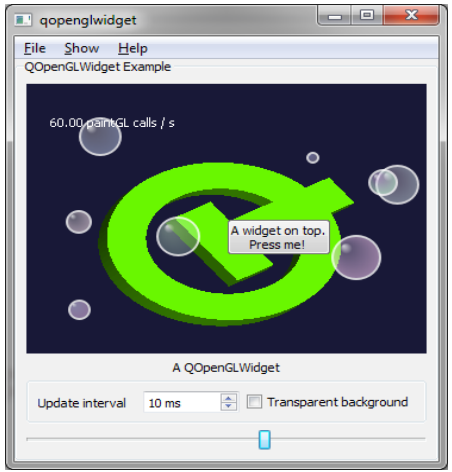
A QWidget with two QOpenGLWidgets running on i.MX6 Sabre LITE on eglfs

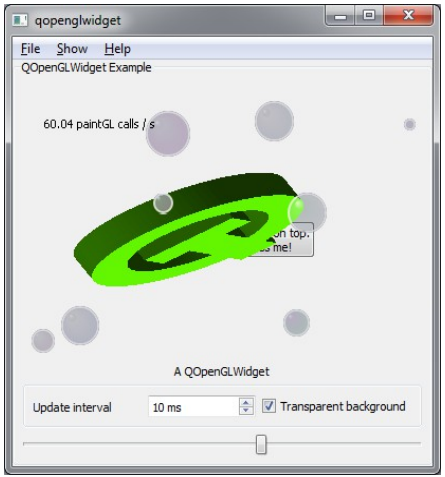


A Qt Quick 2 scene embedded into a widget UI using QQuickWidget

Composition

- Does this mean all widget apps require OpenGL from now on? No.
- Does this support multisampling? Yes.
- Qt::WA_AlwaysStackOnTop





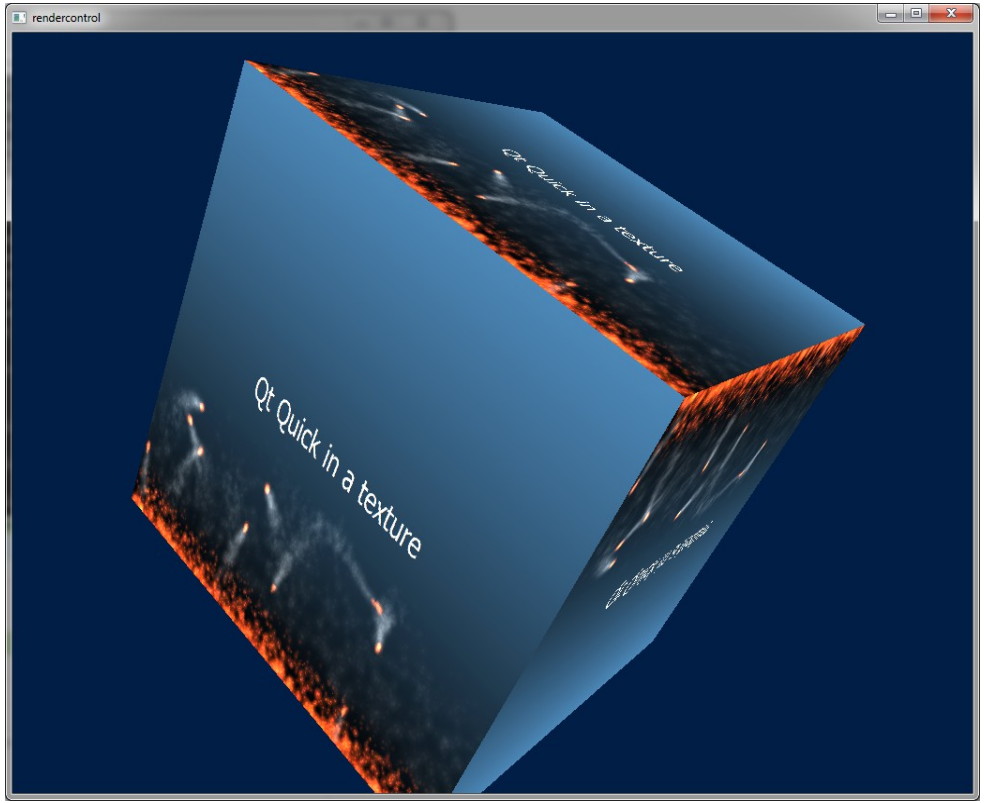
QQuickWidget API

- Mirrors QQuickView
- Except that it is a QWidget

QQuickRenderControl

- ~~QQuickWidget~~
- QQuickRenderControl
- QOpenGLWidget
- QOpenGLWindow
- QRasterWindow

QQuickRenderControl



QQuickRenderControl

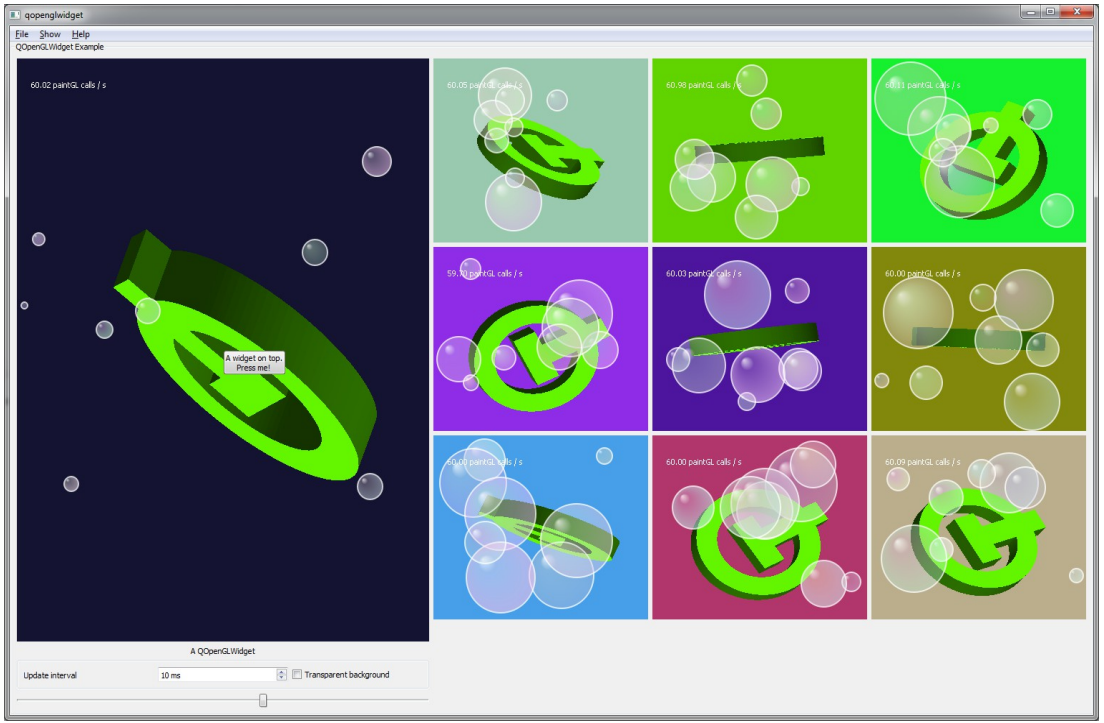
- No on-screen QQuickWindow.
- The rendering of the scene is redirected into a FBO.
- Texture can be used in arbitrary ways. No costly readbacks.
- Need to send input to Quick? QCoreApplication::sendEvent()
- Integrate Quick into external engines, not the other way round.

QOpenGLWidget

- ~~QQuickWidget~~
- ~~QQuickRenderControl~~
- QOpenGLWidget
- QOpenGLWindow
- QRasterWindow

Qt 5.4

- Back to widgets. No more ~~boring~~ QML stuff.
- QOpenGLWidget is here
- R.I.P. QGLWidget



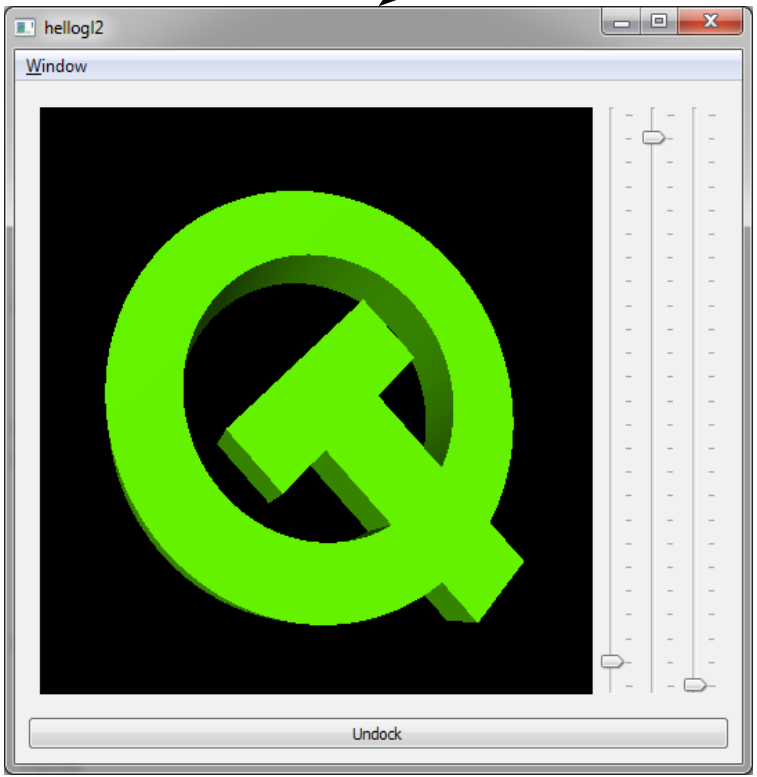
10 QOpenGLWidget instances in one top-level window

QOpenGLWidget

- Basic API is familiar
- Can open a QPainter on it
- Utilities like bindTexture() are gone → QOpenGLTexture

```
class Widget : public QOpenGLWidget, protected QOpenGLFunctions
{
public:
    void initializeGL() { initializeOpenGLFunctions(); ... }
    void resizeGL(int w, int h) { ... }
    void paintGL() { ... }
```

QOpenGLWidget



Window 00050170 "hellogl2" Qt5QWindowIcon
Window 0005016C "hellogl2" Qt5QWindowGLOwnDCIcon

One visible native window only, unlike QGLWidget.

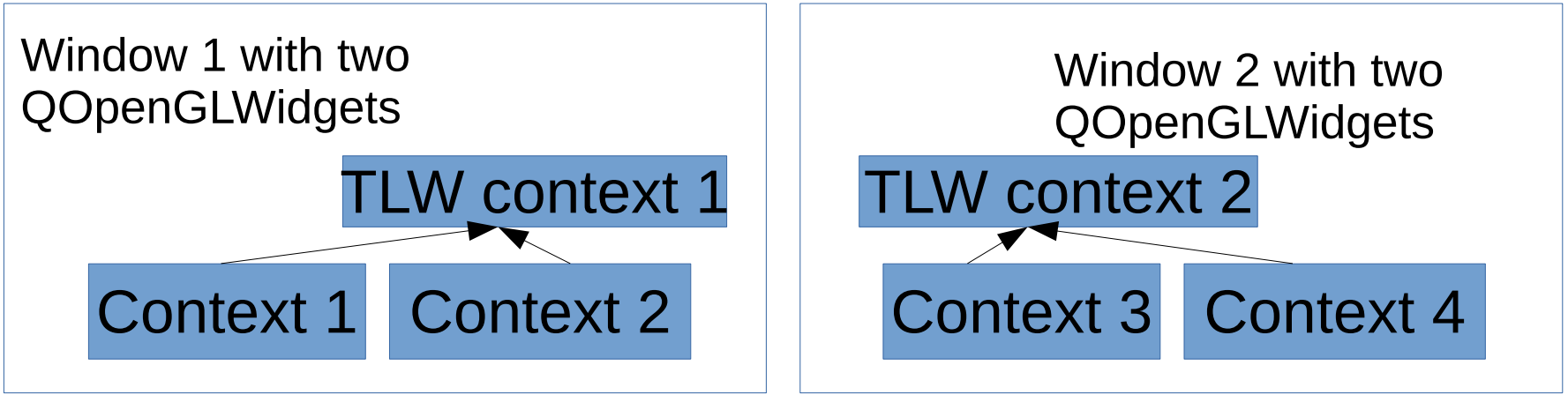
Animation

- ~~**QTimer::singleShot(16, m_legacyQGLWidget, SLOT(updateGL()))**~~
- Swap interval defaults to 1 since Qt 5.3
- QSurfaceFormat::setSwapInterval()
- Schedule repaints with update() and rely on vsync
- Timers with small intervals can be useful, but know what you are doing

Formats and contexts

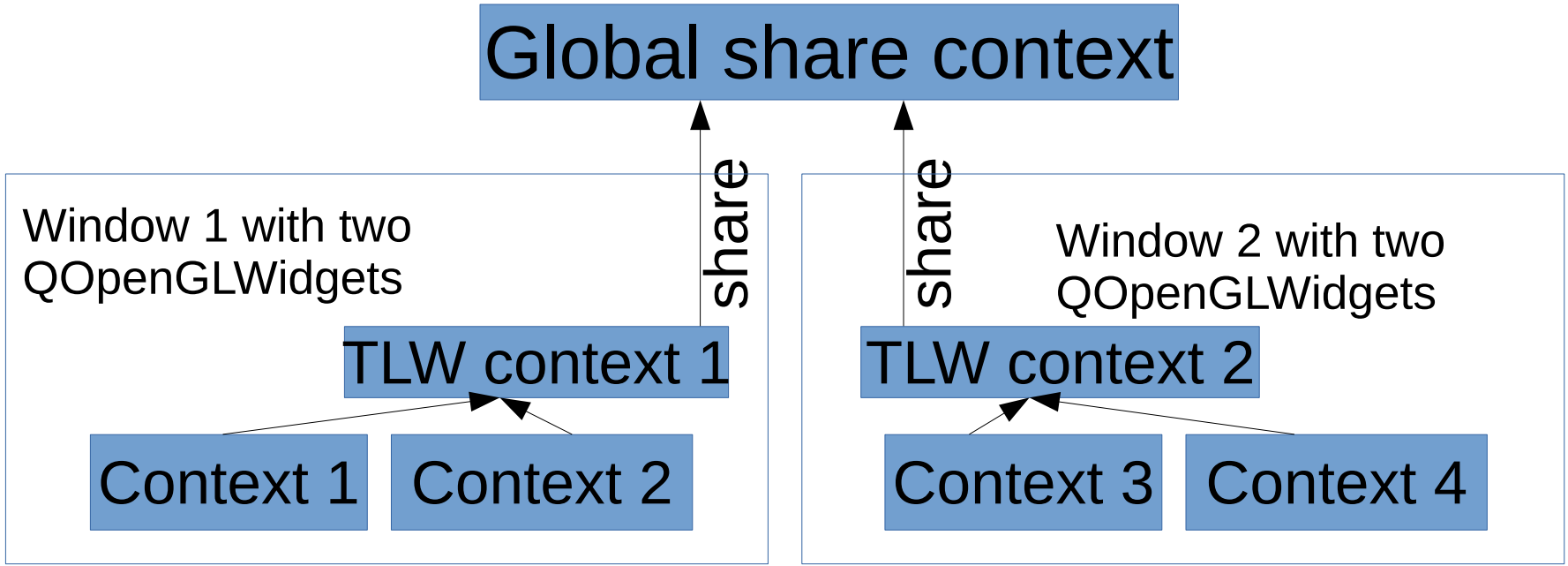
- QSurfaceFormat::setDefaultFormat()
- Easy to request OpenGL 3+ or core profile for everything in the app
- The new compositing architecture relies heavily on multiple contexts and resource sharing.
- QOpenGLWidget comes with documentation. Use it.

Formats and contexts, cont.



Formats and contexts, cont.

- Qt::AA_ShareOpenGLContexts



High DPI screens

- OpenGL operates in pixel dimensions on retina screens.
- *new QOpenGLFramebufferObject(w->devicePixelRatio() * w->size())*
- Qt takes care of all the internal framebuffers and viewports. The rest is up to you.
- Watch out for screen changes
  - screenChanged(QScreen*) signal for QWindow
  - ScreenChangeInternal event for QWidget

QOpenGLWindow

- ~~QQuickWidget~~
- ~~QQuickRenderControl~~
- ~~QOpenGLWidget~~
- QOpenGLWindow
- QRasterWindow

QWindow

- Enough of widgets,
  I don't want your push buttons.
- QWindow is fine for you then.
- Lightweight and powerful, but...

```
class MyWindow : public QWindow
{
public:
    MyWindow() {
        setSurfaceType(QSurface::OpenGLSurface);
        QSurfaceFormat format; ...
        m_context.setFormat(format);
        m_context.create();
    }
    void exposeEvent(QExposeEvent *) {
        if (isExposed())
            render();
    }
    void resizeEvent(QResizeEvent *) { ... }
    void render() {
        m_context.makeCurrent(this);
        QOpenGLFunctions *f = m_context.functions();
        f->glClear(GL_COLOR_BIT | GL_DEPTH_BUFFER_BIT);
        ...
        m_context.swapBuffers(this);
    }
```

QOpenGLWindow

```
class MyWindow : public QOpenGLWindow {
    void resizeGL(int w, int h) { ... }
    void paintGL() {
        QOpenGLFunctions *f = context()->functions();
        f->glClear(GL_COLOR_BIT | GL_DEPTH_BUFFER_BIT);
        // issue some native OpenGL commands
        ...
        QPainter p(this);
        // draw using QPainter
        ...
        // animate continuously: assume blocking swap and just schedule an update
        update();
    }
};
```

QRasterWindow

- ~~QQuickWidget~~
- ~~QQuickRenderControl~~
- ~~QOpenGLWidget~~
- ~~QOpenGLWindow~~
- QRasterWindow

QRasterWindow

- QOpenGLWidget has a little brother: QRasterWindow
- Exactly what the name suggests. Nothing more, nothing less.

```
class HelloWindow : public QRasterWindow
{
    void paintEvent(QPaintEvent *) {
        QPainter painter(this);
        painter.fillRect(0, 0, width(), height(), Qt::white);
        painter->drawText(QRectF(0, 0, width(), height()), Qt::AlignCenter, QStringLiteral("Hello world"));
    }
};
```

Summary

- ~~QQuickWidget~~
- ~~QQuickRenderControl~~
- ~~QOpenGLWidget~~
- ~~QOpenGLWindow~~
- ~~QRasterWindow~~

Qt Quick 2 application	QQuickView or QQmlApplicationEngine
OpenGL without any need for Qt widgets	QOpenGLWindow
Custom QPainter-based drawing without Qt widgets	QRasterWindow
OpenGL content in a widget-based application	QOpenGLWidget
Qt Quick 2 content in a widget-based application	QQuickWidget
Qt Quick 2 content used as a texture in a custom OpenGL renderer	QOpenGLWindow + QQuickRenderControl

What was left out

- Dynamic OpenGL implementation loading on Windows
- Adoption of existing native OpenGL contexts

Thank you!

www.qt.io

www.qt.io

See you there!