GUIs: Coming to Uncommon Goods Near You

Jason Kridner

With the “smartphone effect”, proliferating in many applications outside the consumer space, it’s become apparent that slick graphical user interfaces (GUIs) sell. In this session, you’ll learn how to quickly develop a GUI for your product using Linux. Learn about cool tools such as Qt Creator and the entire Qt toolset. Soon, your basic washing machine control panel could be just as excitiing as a smartphone!

Agenda

- Why fancy GUIs everywhere?
  - Which one to choose?
- Introduction to QT
  - Hello World
  - The QT Framework
- Exploring the examples/demos

The pinch effect - user demand

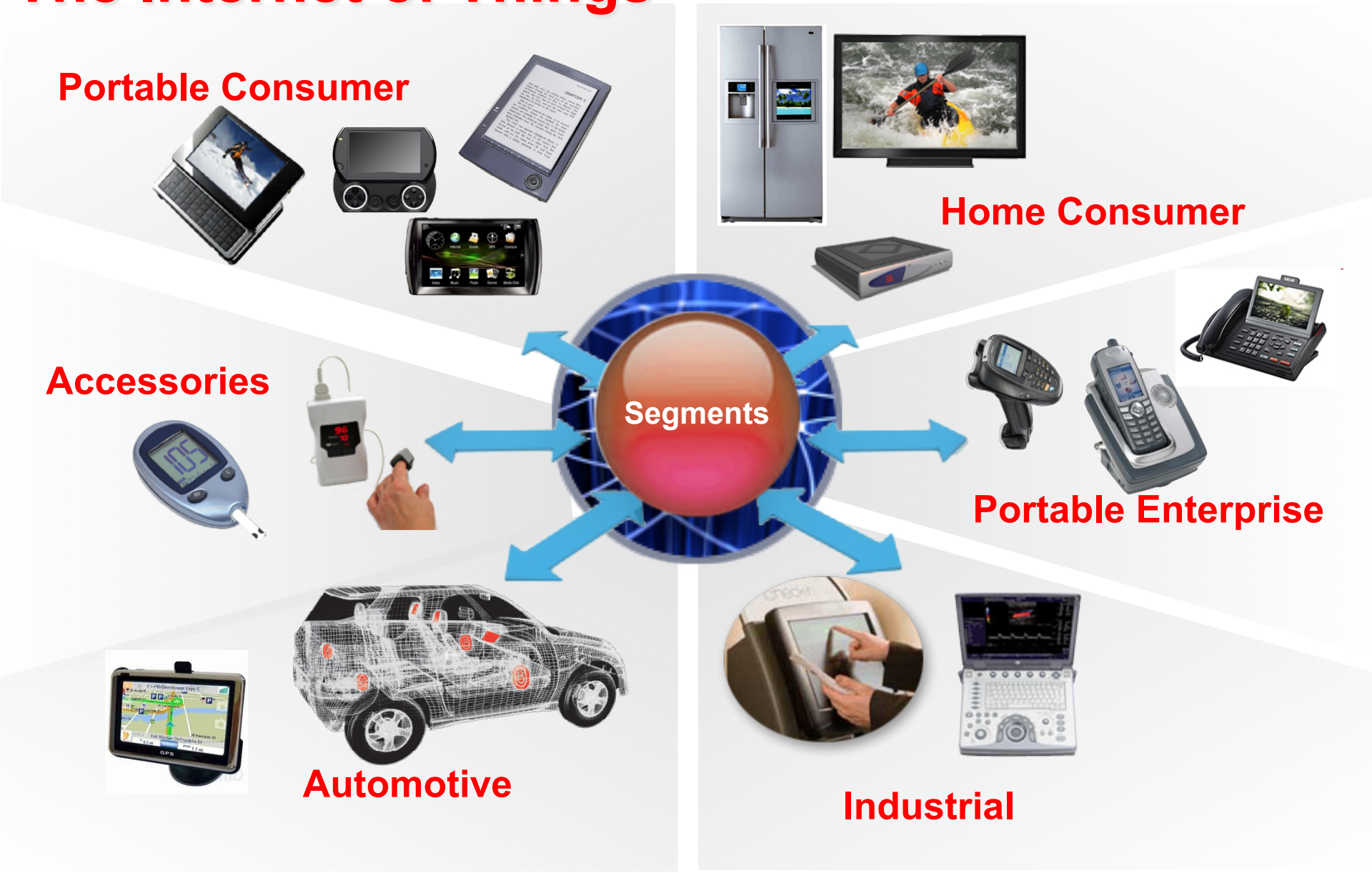
- **QNX's Andy Gryc**, senior product marketing manager for QNX Software Systems says
  - He's seen a trained engineer "forget" how to operate an oscilloscope and attempt to use the pinch-and-spread **gesture to zoom** into a scope trace.

- **Beckhoff's McAtee** takes it further.
  - "[If you] combine [multi-touch] functionality with wide format 24-in. screens, device vendors and machine designers would be able to **remove all physical push buttons** from the panel, allowing the user to manage every machine function directly on the touchscreen. This would permit easy scrolling and zooming through dashboards and menus, beyond the capabilities of conventional touchscreen technology."

- **Fujitsu's Bruce DeVisser**, product marketing manager for touch input
  - technologies have crossed over into the industrial space. "**Haptic feedback**, embodied as a vibration of the touch panel (like how a cell phone vibrates), is very useful for noisy industrial environments". A display in black mode (power-saving or screen-saver state) is unappealing to [consumer] users if it is covered with fingerprints.

Source: http://m.controleng.com

The Internet of Things



Some GUI options

HTML

JavaScript

Java (C)

C++ (JavaScript)

Skill required

Performance

More on GUI options

HTML

Closures (DBUS/REST/…)	Activity/Intent	Signals/Slots
Browser	Phone/Tablet/…	Cross platform

Qt – Getting Started

The TI SDK setup

- Install the Sitara SDK on your host PC running Ubuntu
- Ensure that the PATH environment variable contains qmake
  - source $(SDK_HOME)/linux-devkit/environment_setup

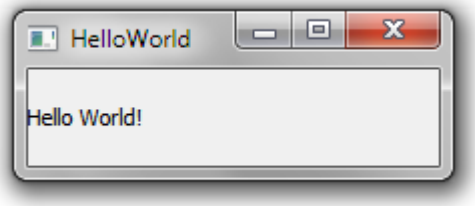
“Hello World!”

- Create a working directory “helloworld”
- Create a C++ source file “main.cpp” using your favorite editor with the following contents

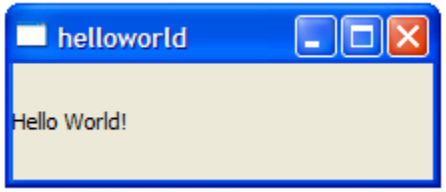
```
#include <QApplication>
#include <QLabel>

int main(int argc, char **argv)
{
    QApplication app(argc, argv);

    QLabel label("Hello World!");
    label.show();

    return app.exec();
}
```

Running “Hello World!”

- Run qmake inside the helloworld directory to create a project file
  - qmake –project



- Run qmake again to create a Makefile from helloworld.pro
  - qmake



- Run make to build the application
  - make
- Application is built and ready in debug/ directory. Copy executable to your filesystem on your target and run.

Running Supplied Demo Applications

- There are over 300 demo and example applications supplied in the SDK.
  - They come from the QT SDK and are not supported by TI
  - Wide variety of applications. The same application from QT Demo.
  - The example application already contain a project file.
  - Found at $(SDK_HOME)/linux-devkit/arm-arago-linux-gnueabi/usr/bin/qtopia
    - demos
    - examples
- To build the supplied Demo application on your host
  - Run qmake to create a Makefile from project file *.pro
    - qmake
  - Run make to build the application
    - make
  - The application is built and ready in debug/ directory. Copy executable to your filesystem on your target and run.

Example Applications

- Matrix GUI Application Launcher provided in the SDK
  - Built with QT utilizing Webkit.

Matrix GUI Development - Components

- Menus / Submenus / Description
  - Each Menu,Submenu or Description page is generated by 1 HTML file
- HTML files
  - Each HTML file contains a header and references up to 8 or 12 icons
  - Each icon is associated with an submenu or an application
- Icons
  - 96x96 pixel images representing the application
  - Blank icons available for future development
- Applications
  - Each application is associated with an icon

Cascading Style Sheets with HTML

- Matrix GUI contains one Cascading Style Sheet (CSS) – **matrix.css**
  - Each HTML file reads in matrix.css
  - matrix.css controls the look and feel of all the Matrix GUI HTML pages
    - Automatically controls spacing of the icons and text labels
    - Automatically centers the text labels underneath the icons
    - Supports wQVGA (480x272) up to 1080p resolution (1920x1080)

Top 15 lines of matrix.css

```
{color: #ffffff;} /* Default all text to white */

/* Set the background color to black */
body {background-color: #000000;}

/* This secton controls both the icon image and the text label together */
div.object
{
  text-align: center;
  float: left;
  background-color:#000000;
  width: 25%;
  height: 30%;
}
```

Cascading Style Sheets in Action

wQVGA – 480x272

VGA – 640x480

- Matrix GUI displayed on two different LCD displays with different resolutions

- Only requires minor changes to the matrix.css HTML cascading stylesheet
  - Scale icons down to 64x64 for wQVGA / remain native 96x96 for VGA
  - Decrease font size for wQVGA / increase font size for VGA
  - wQVGA - each icon 45% of display in height / VGA each icon 30% height

Matrix GUI – Adding a new application

- The HTML below represents one application associated with one Icon.
- To add an additional application simply cut and paste this HTML segment and fill in the <red> fields

```
<div class="object">
    <object type="application/x-matrix" >
      <param name="iconName" value= <"icon path"> />
      <param name="appName" value= <"application path"> />
      <param name="appParameters" value= <"parameters"> />
    </object>
    <div class="desc"> <"Label"> </div>
</div>
```

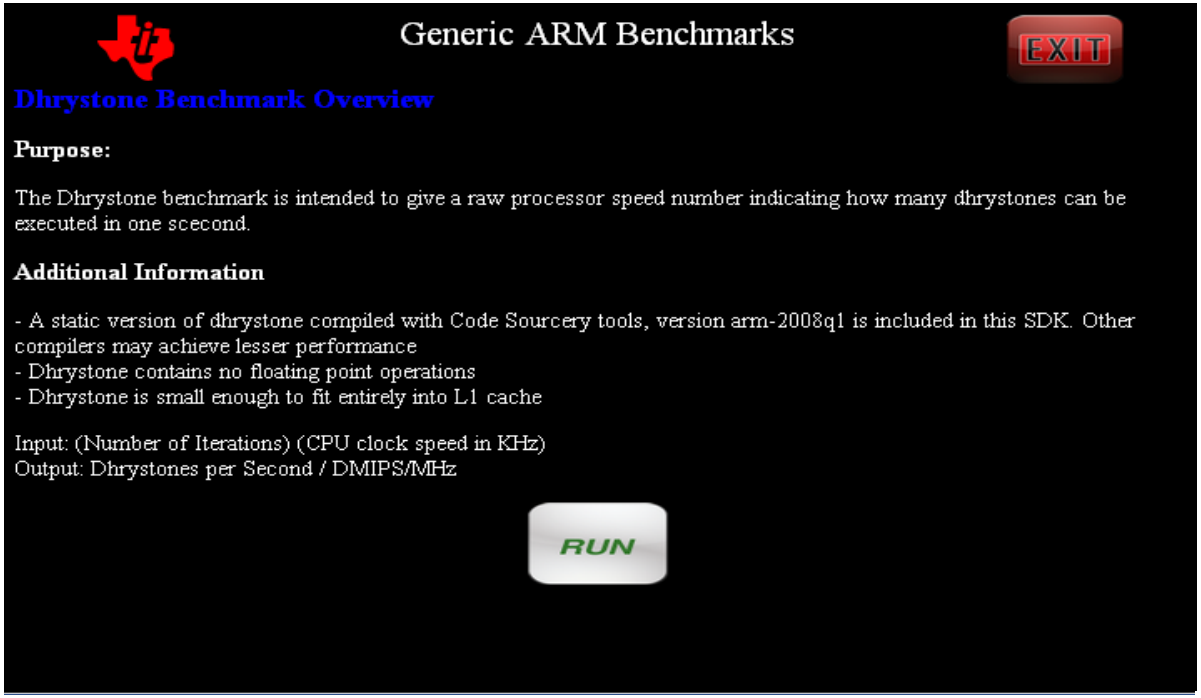
- iconName, appName, and desc fields are manditory
- appParameters and any other fields are optional

Matrix GUI – HTML Header

```
<body>
  <div class="topBar">
    <object type="image/svg+xml” data="/usr/share/matrix/
  images/tex.svg" >
    <img src="/usr/share/matrix/images/tex.svg" />
  </div>
  <div id="header">Matrix Application Launcher p1</div>
  <div class="topBar">
    <object type="application/x-matrix" >
      <param name="iconName" value="/usr/share/matrix/
  images/exit-icon.png" />
      <param name="appName" value="Close" />
    </object>
  </div>
  <div class="topBar">
</div>
```

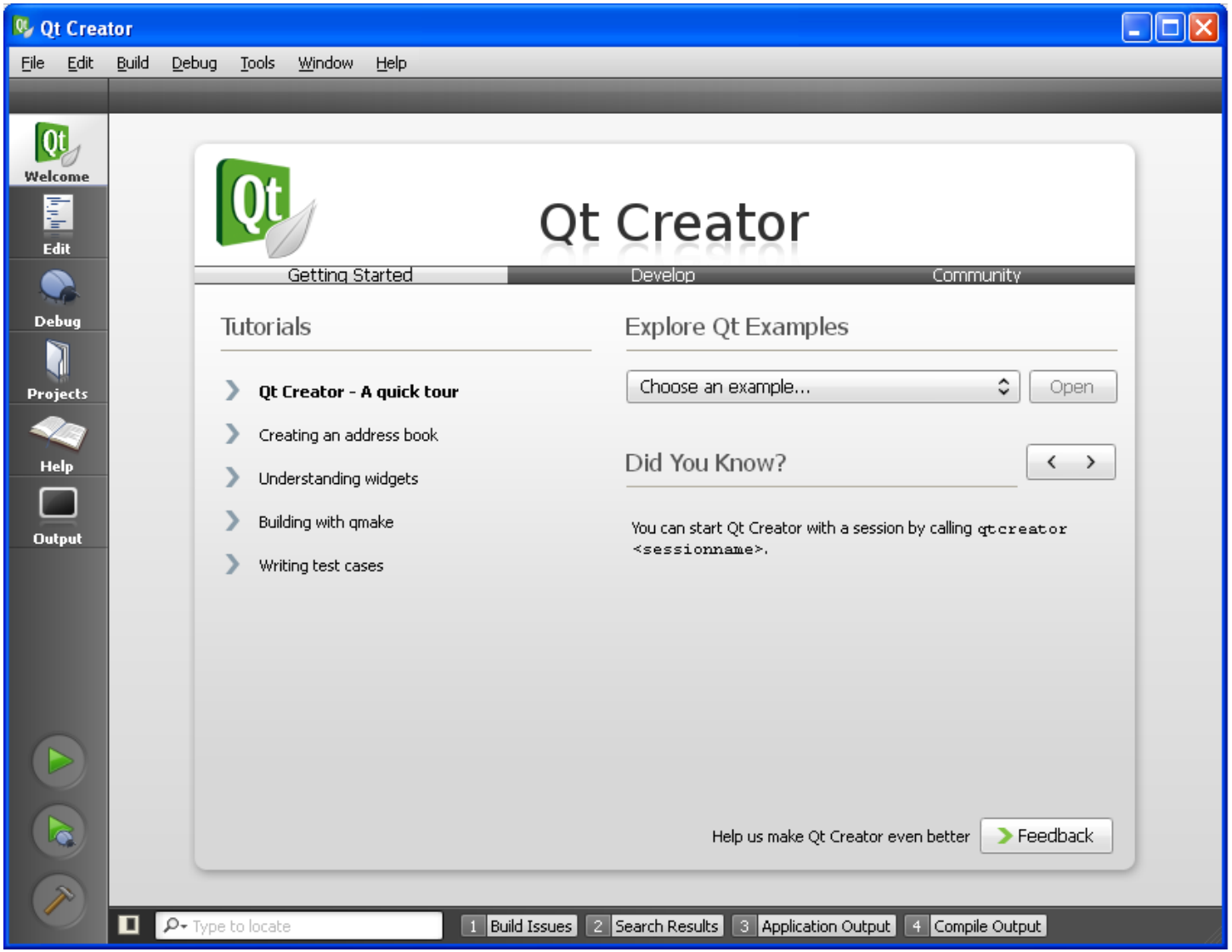
Application Description Pages

- Applications can optionally have a description page
- Descriptions pages:
  - Add additional info
  - Provide setup steps
  - Point out valuable features
- Description mode is defaulted to on, but can be disabled
- Push ARM
- Push Dhrystone
- Push Run

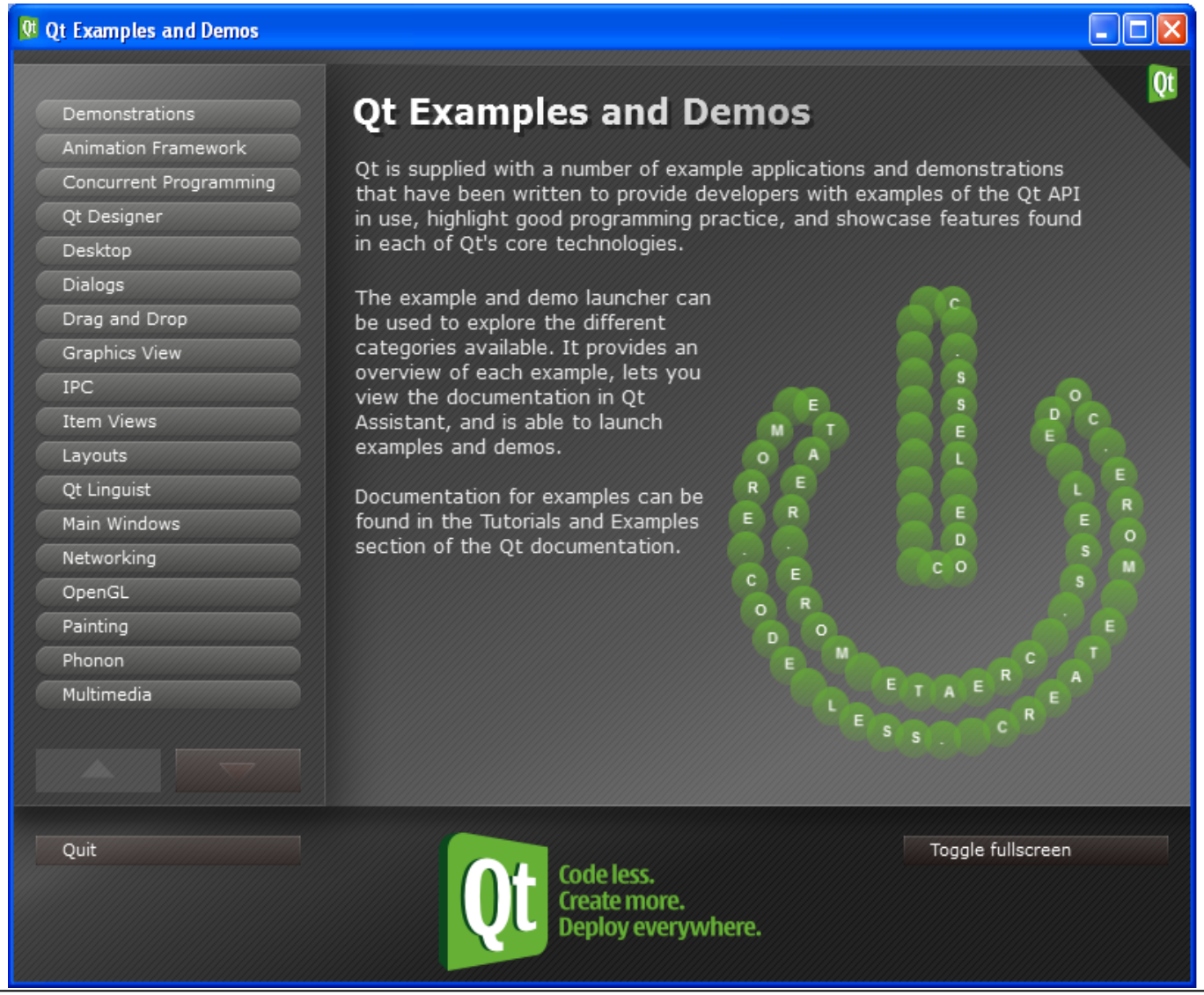


- When you push the icon to run the application, if a description is available it pops up.

QT Creator – Development tools



QT Demo – Application Example Projects



Introduction to Qt

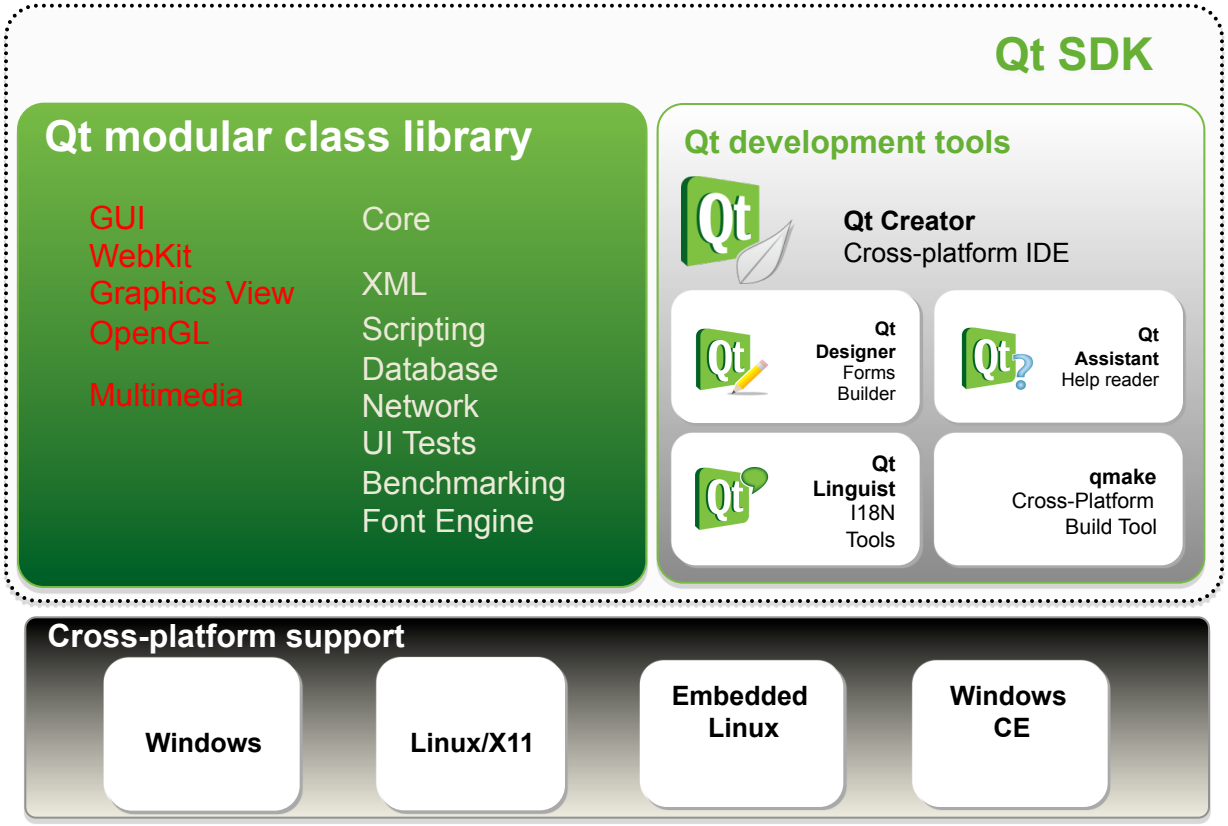
What's Qt?

- Cross platform application / UI framework
- Portable - Same API across desktop and embedded OS
- Supported on various platforms

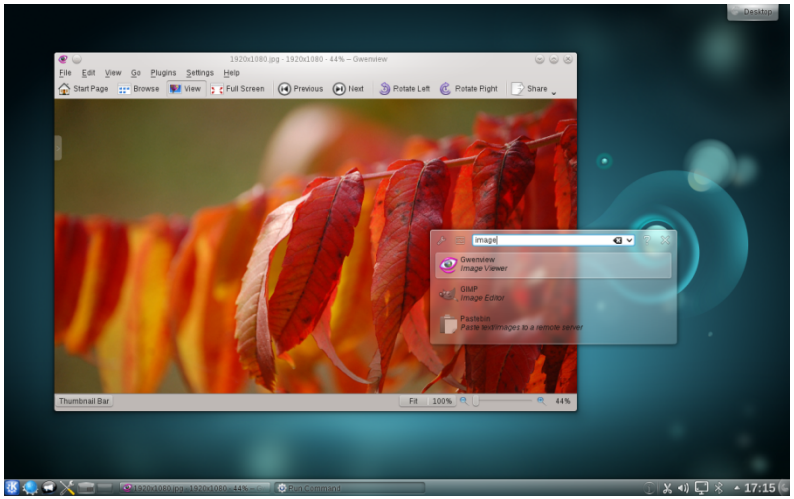
Desktop OS	Embedded OS
Windows	Embedded Linux
Linux/X11	Symbian
Mac OS	Meego / Maemo

- External ports being developed for:
  - Android
  - iPhone
  - Wayland
  - webOS, OpenSolaris, Amiga, OS/2, …

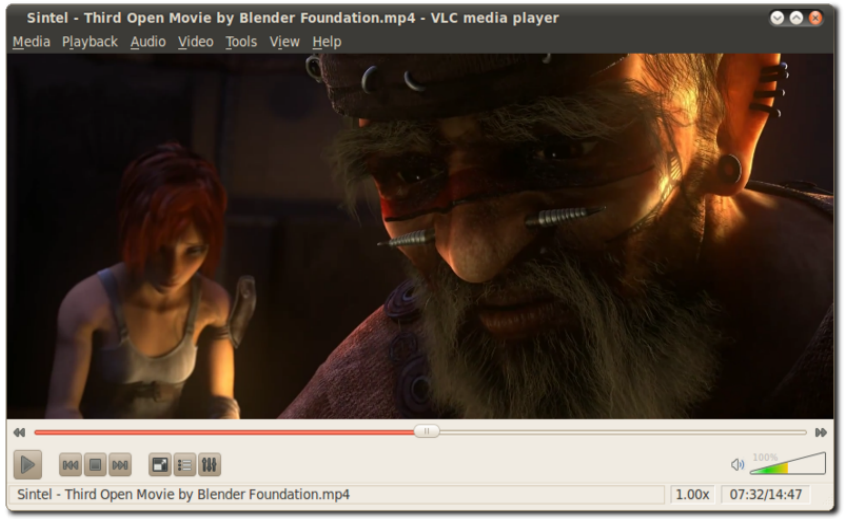
- Qt is cross-platform application and UI framework.
- Qt provides a well defined API that can make development quick and easy.
- Webkit
  - Well accepted open source web browser
  - Rapidly create real-time web content and services
  - Use HTML and Java Script integrated in native code
- 3D Graphics with OpenGL and OpenGL ES
  - Easily incorporate 3D Graphics in your applications
  - Get maximum graphics performance
- Multithreading support
- Network connectivity
- Advanced GUI development



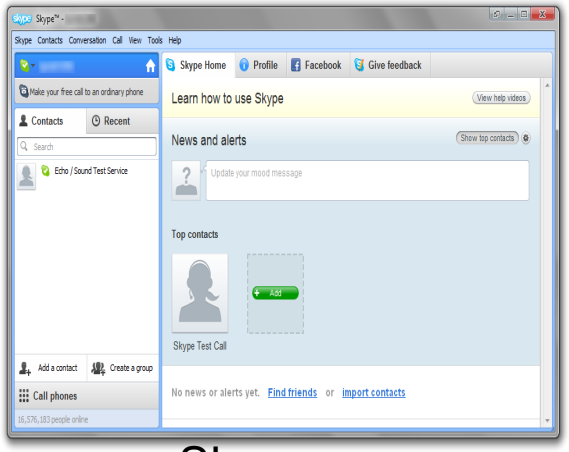
Qt usage – these and much more …



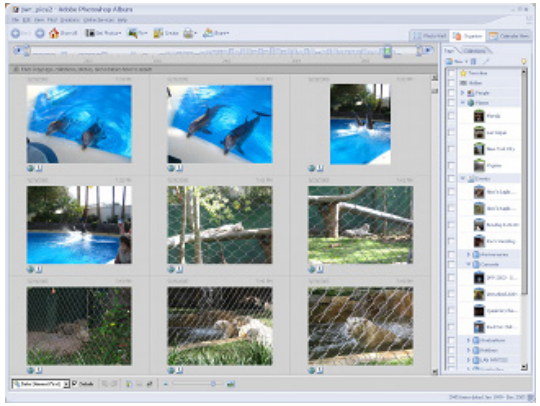
KDE



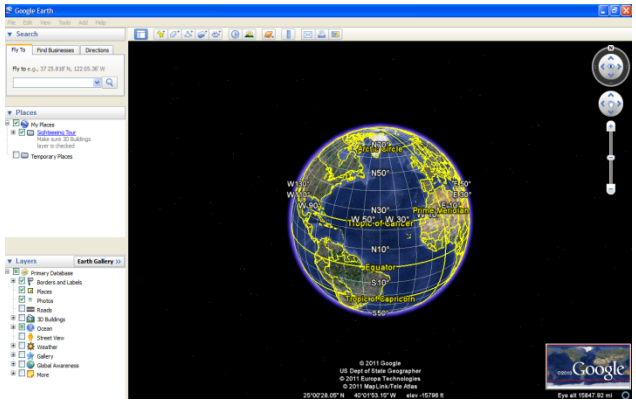
VLC Media Player



Skype



Adobe Photoshop Album



Google Earth

Webkit applications

- Webkit
  - Google Chrome
  - Safari
  - Experimental Kindle browser
  - Matrix GUI

Qt – Brief History

1994	• Haavard Nord & Eirik Chambe-Eng incorporated Quasar • Became Trolltech
1996	• Qt 1.0 released • Supported on Windows, Unix/X11 • Decision to use Qt for developing KDE
2001	• Qt 3.0 released • Supported on Windows, Linux, Mac OS, Embedded • Open Source license
2005	• Qt 4.0 released • Performance optimized • Vast application classes
2008	• Nokia acquires Trolltech • Port for Symbian S60 platform
2011*	• Nokia announce strategic partnership with Microsoft • Digia acquires Qt's commercial licensing and support

Qt Licensing

	Commercial	LGPL v2.1	GPL v3.0
License Cost	License fee charged	No license fee	No license fee
Must provide source code for changes to Qt	No, modifications can be closed	Source code must be provided	Source code must be provided
Can create proprietary application	Yes – No source code must be disclosed	Yes, in accordance with the LGPL v2.1 terms. (Must dynamically link.)	No, applications are subject to the GPL and source code must be made available
Updates Provided	Yes, immediate notice sent to those with a valid support and update agreement	Yes, made available	Yes, made available
Support	Yes, to those with a valid support and update agreement	Not included but available separately for purchase	Not included but available separately for purchase
Charge for Runtimes	Yes, for some embedded uses	No	No

Qt Releases

Qt Release	URL
Qt SDK for Windows	http://get.qt.nokia.com/qtsdk/qt-sdk-win-opensource-2010.05.exe
Qt SDK for Linux	http://get.qt.nokia.com/qtsdk/qt-sdk-linux-x86-opensource-2010.05.1.bin
Qt Framework for Embedded Linux	http://get.qt.nokia.com/qt/source/qt-everywhere-opensource-src-4.7.2.tar.gz

- Qt SDK contains the following:
  - Qt Framework - C++ classes that form the building blocks of Qt
  - Qt Creator - Cross platform IDE for developing Qt applications
  - Qt Designer - Easy GUI designer to build layout and forms
  - Qt Linguist - Tools that aid translation and internationalization
  - Qt Assistant - Documentation and help system

Qt Framework & Internals

Qt - Application development flow

Build Qt for target

Create .pro file

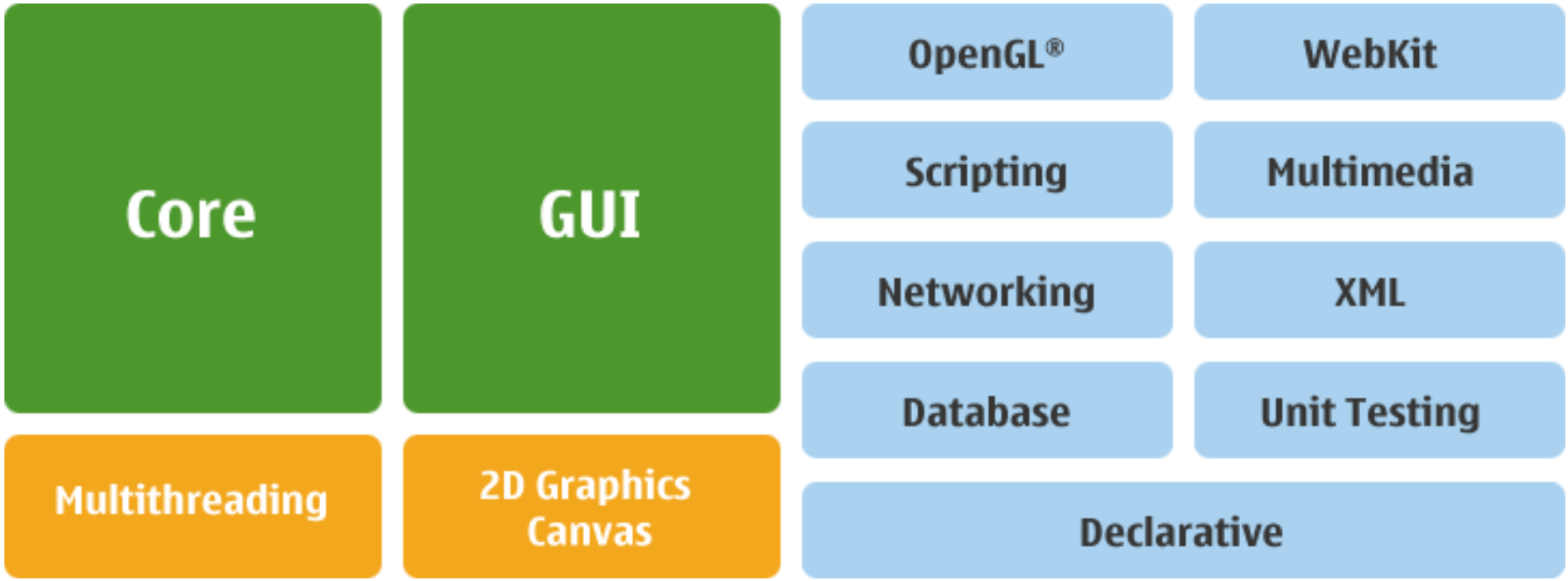
Design the UI in Qt designer

Add necessary event handlers

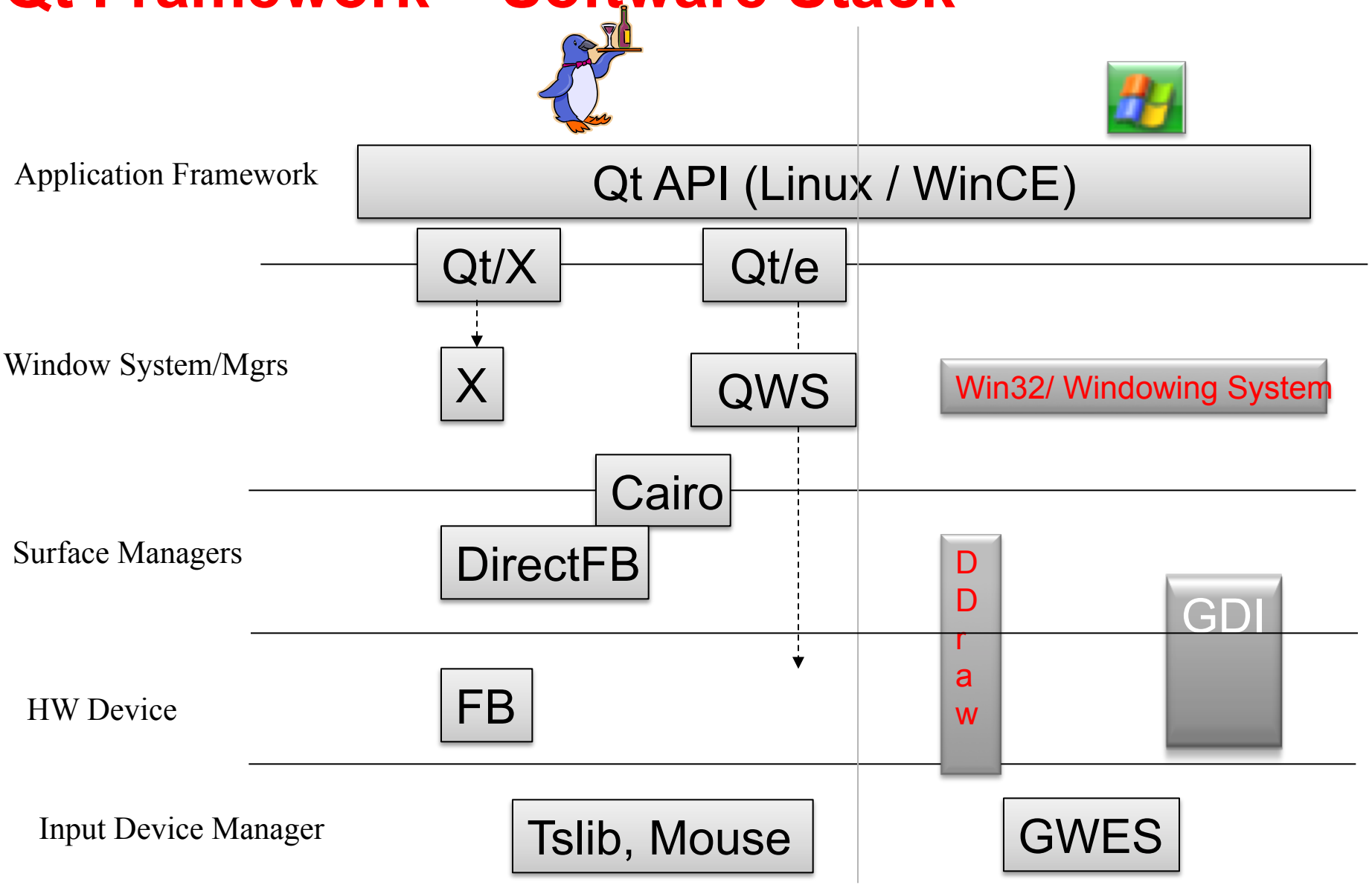
Add necessary application code

Build & Install

Qt Framework – Application Classes



Qt Framework – Software Stack



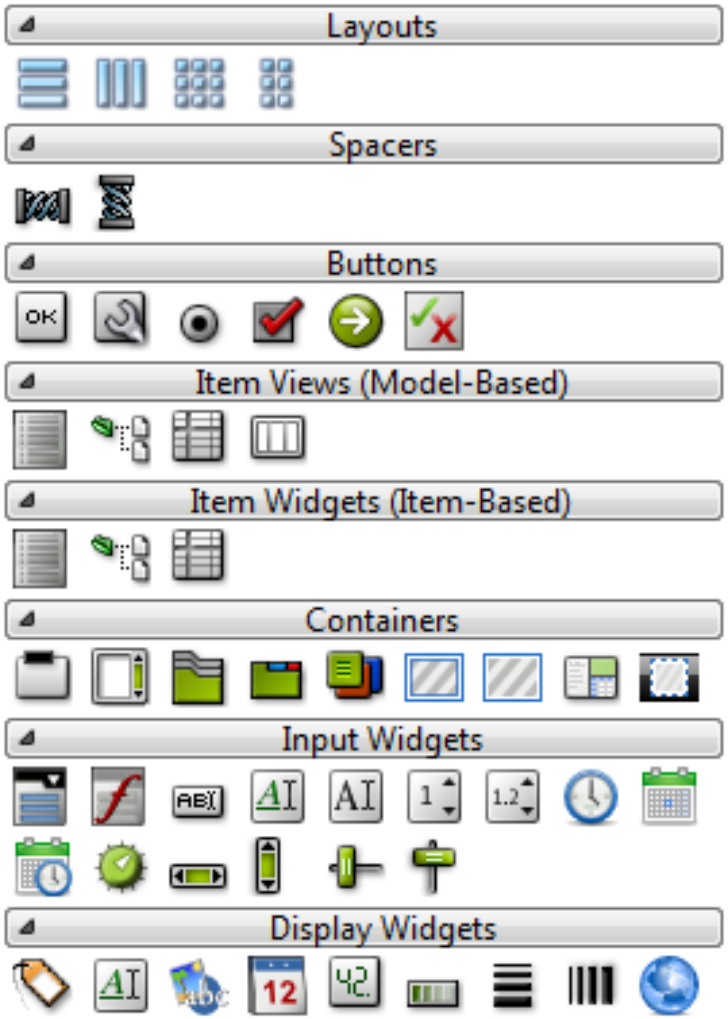
Widgets (1/2)

- Qt UI framework is based on widgets
- Widgets respond to UI events (key presses/ mouse movements), and update their screen area
- Each widget has a parent, that affects its behavior, and is embedded into it
- Most Qt classes are derived from QWidget
  - Ex, QGLWidget, QPushbutton …

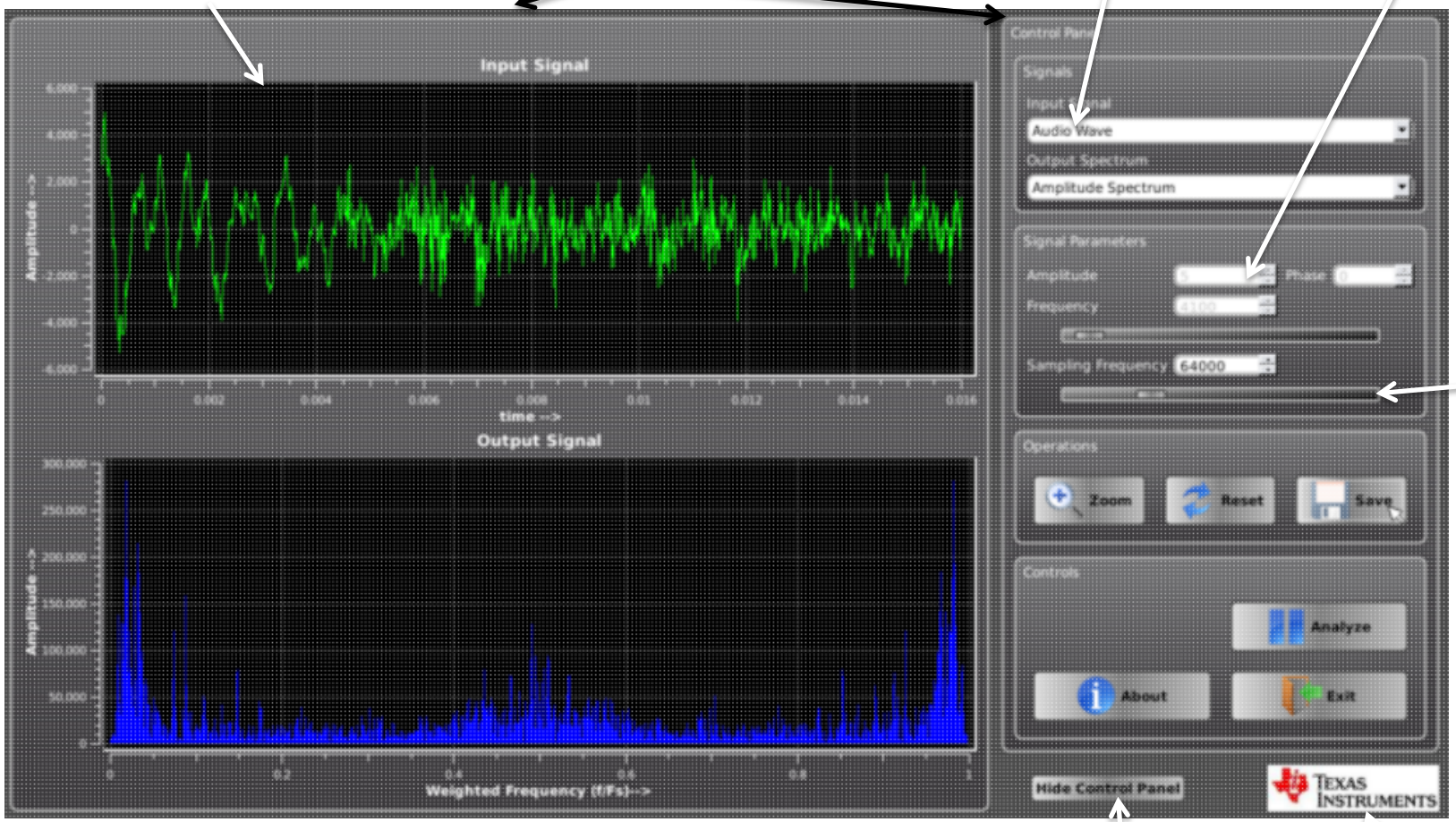
    *QPushButton * myButton = new QPushButton(…);*

    myButton->doSomethingAPI();

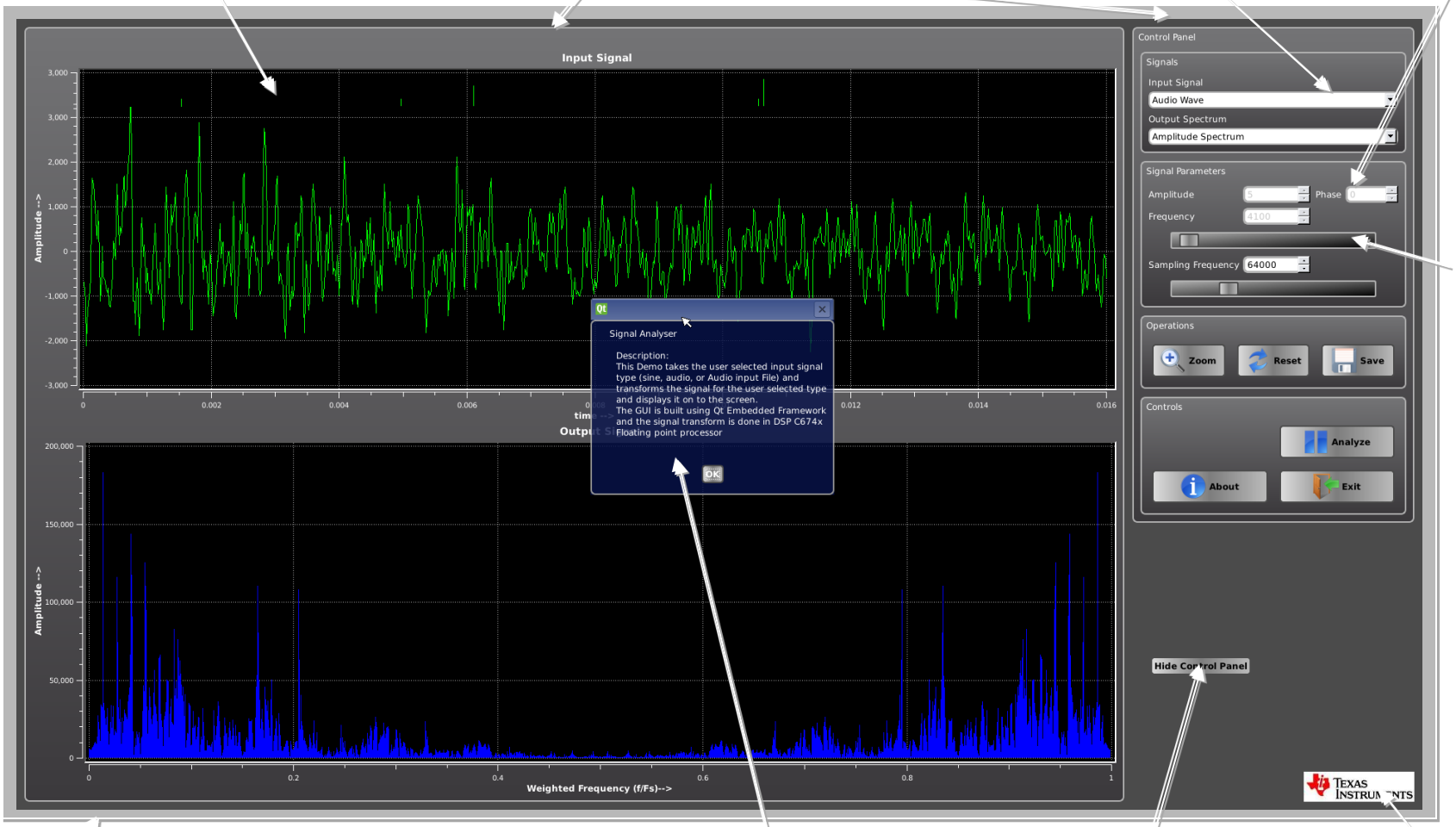
- Refer to online documentation at
  - http://doc.qt.nokia.com/4.6/qwidget.html
  - Tip – Documentation is arranged using class names.



Widgets (2/2)


QWTPlotCurve
QHBoxLayout
QComboBox
QDoubleSpinBox
QMainWindow
QSlider
QPushButton
Qlabel
Input Signal
Output Signal
time -->
Weighted Frequency (f/Fs)-->
Control Panel
Signals
Input Signal
Audio Wave
Output Spectrum
Amplitude Spectrum
Signal Parameters
Amplitude
Phase
Frequency
Sampling Frequency
64000
Operations
Zoom
Reset
Save
Controls
Analyze
About
Exit
Hide Control Panel
Texas Instruments

Widgets (2/2)


QWTPlotCurve
QHBoxLayout
QComboBox
QDoubleSpinBox
QSlider
Input Signal
Amplitude -->
Control Panel
Signals
Input Signal
Audio Wave
Output Spectrum
Amplitude Spectrum
Signal Parameters
Amplitude
Phase
Frequency
Sampling Frequency
64000
Operations
Zoom
Reset
Save
Controls
Analyze
About
Exit
Signal Analyser
Description:
This Demo takes the user selected input signal type (sine, audio, or Audio input File) and transforms the signal for the user selected type and displays it on to the screen.
The GUI is built using Qt Embedded Framework and the signal transform is done in DSP C674x Floating point processor
OK
Weighted Frequency (f/Fs)-->
Hide Control Panel
Texas Instruments
QMainWindow
QMessageBox
QPushButton
Qlabel

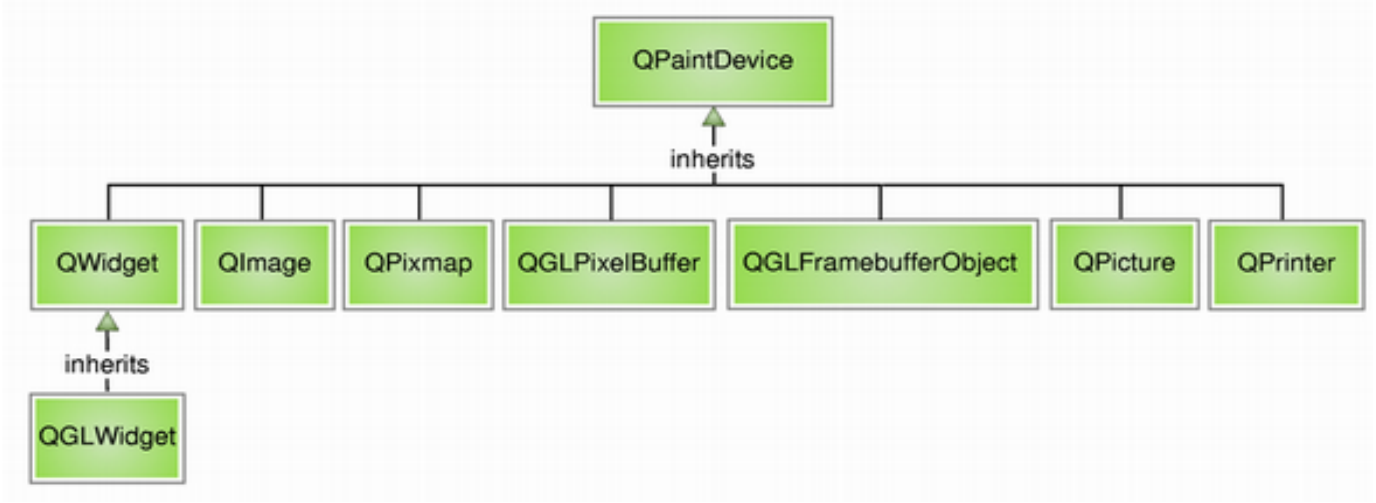
Painting in Qt (1/2)

- QPainter
  - Low level painting API for overriding default painting behavior
  - Uses Pen, Brush, Color to draw
  - Can paint various shapes
    - Point(s)
    - Line(s)
    - Rectangle
    - Ellipse
    - Polygon
    - Arc
    - Polygon
    - Text
    - Image
  - Supports transformations – scale, rotate, translate, shear
  - Paints on a QPaintDevice object

Painting in Qt (2/2)

- QPaintDevice
  - Objects that can be painted by a QPainter using QPaintEngine
  - Could be
    - QWidget
    - QImage
    - QPixmap
    - QGLPixelBuffer
    - QPicture
    - QPrinter
- QPaintEngine
  - Specifies how painting is to be done for a specific device
  - Support for
    - X11
    - CoreGraphics
    - OpenGL
    - Raster Paint

3D graphics in Qt



- Allows 3D operations to be performed in a widget
- As like any widget, QGLWidget operates on a target buffer
- QGLWidget is implemented in src\opengl\qgl.cpp

Graphics View Framework

- Provides a “Canvas” for adding items (QGraphicsItems)
- The QGraphicsView class provides a widget for displaying the contents of a QGraphicsScene
- By default, QGraphicsView provides a regular QWidget for the viewport widget.
  - Can replace default by calling setViewport() with another widget type
- Provides a way to build an UI in an “actual” drawing canvas
  - Ex, concept of “z-depth” of a QGraphicsItem
- To render using OpenGL, simply call:
  - setViewPort(new QGLWidget)

Signals & Slots

- Signal / Slot mechanism provides a functionality similar to setting up “function pointers”
  - Provides better type checking, amongst others
- Example Use-case: Perform blocking/ time consuming activities in separate thread
  - Use paintEvent() to trigger/consume the result of actions happening in parallel (ex. Upload next video frame)
- How to communicate events ?
  - Use SIGNAL/SLOT to communicate event completions
- Usage example for Signal/Slots:
  - “browserlib” app in xgxperf
    - Found in /Xgxperf/browserlib/browserlib.cpp

Using SIGNAL / SLOT

```
Class myClass: public QWidget
{
Q_OBJECT /* Needed for signal/slot mechanism to work at runtime */
public:  …
signals:
        void function1(const QImage &image, double scaleFactor);
};
In thread code,
        emit function1(image, scaleFactor);
```

In Main application, define the actual function::

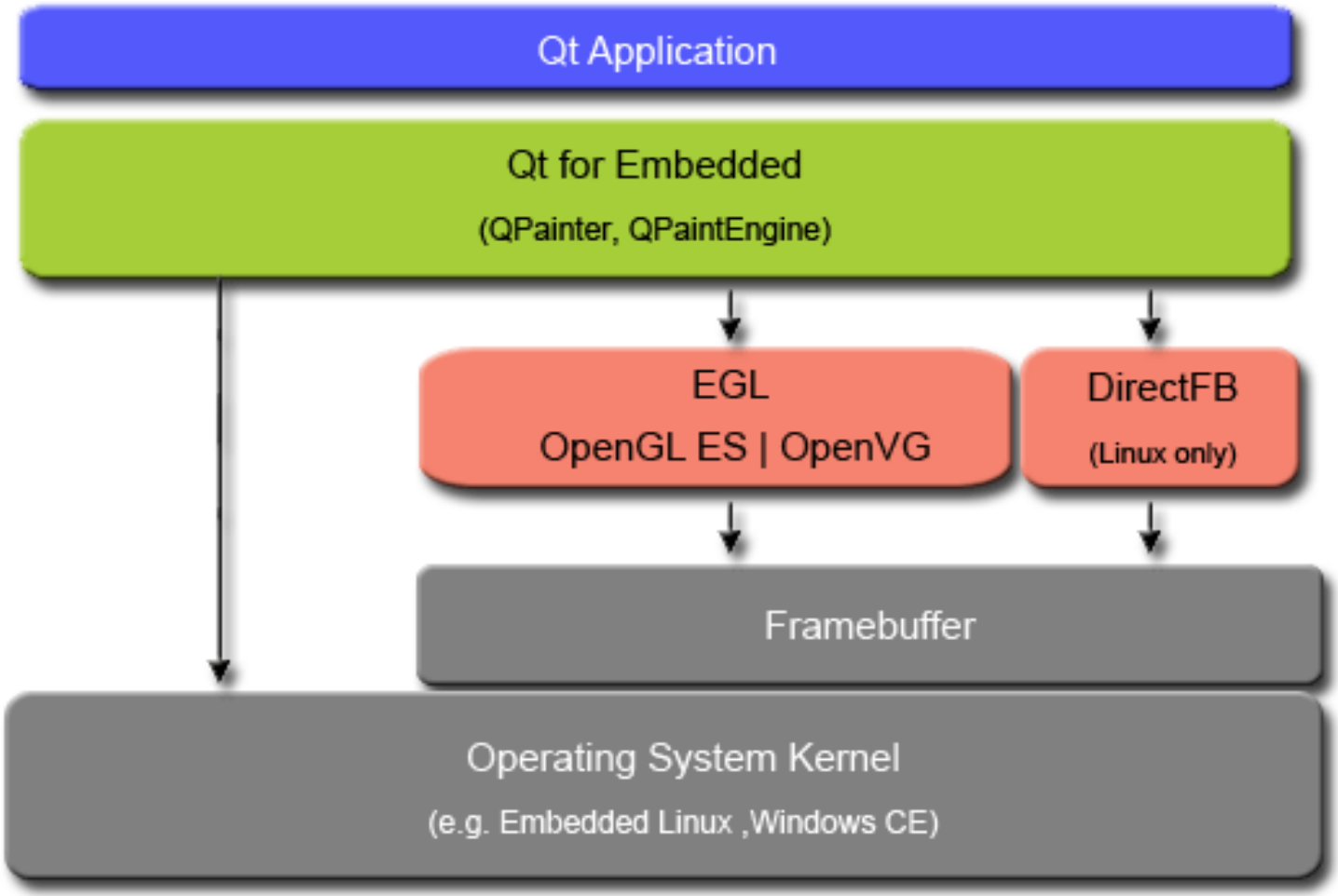
void myWidget::mainWidgetFunction(const QImage &image, double scaleFactor){}

…

And connect the signal and slot:

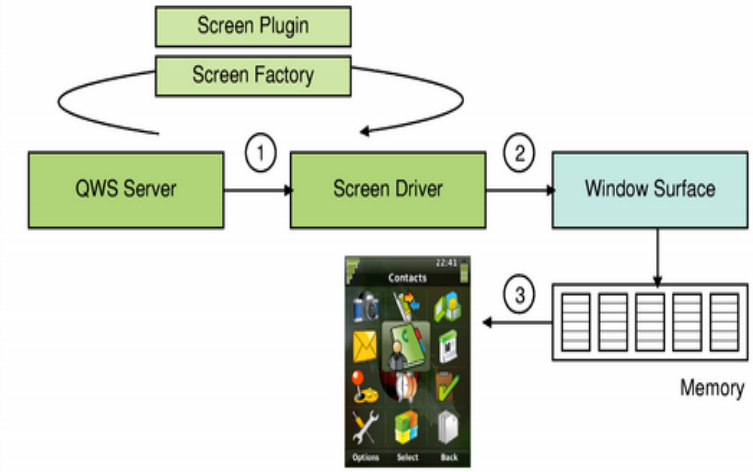
```
connect(&thread, SIGNAL(mainWidgetFunction(const QImage &, double)),
    this, SLOT(function1(const QImage &, double)));
```

Qt/Embedded Linux Pipeline



Screen Driver Architecture

- Specific to Qt/Embedded Linux
- QWS Server loads the screen driver at initialization. Can be specified at run time by “-display <screen driver>”
- QWS supports Linux FB, Virtual FB, VNC, Multi Screen. Default is Linux FB at /dev/fb0
- Qt also supports SGX based powervr screen driver
- Netra supports FBDev driver on Cortex-A8. This internally uses SysLink to communicate with HDVPSS drivers on M3



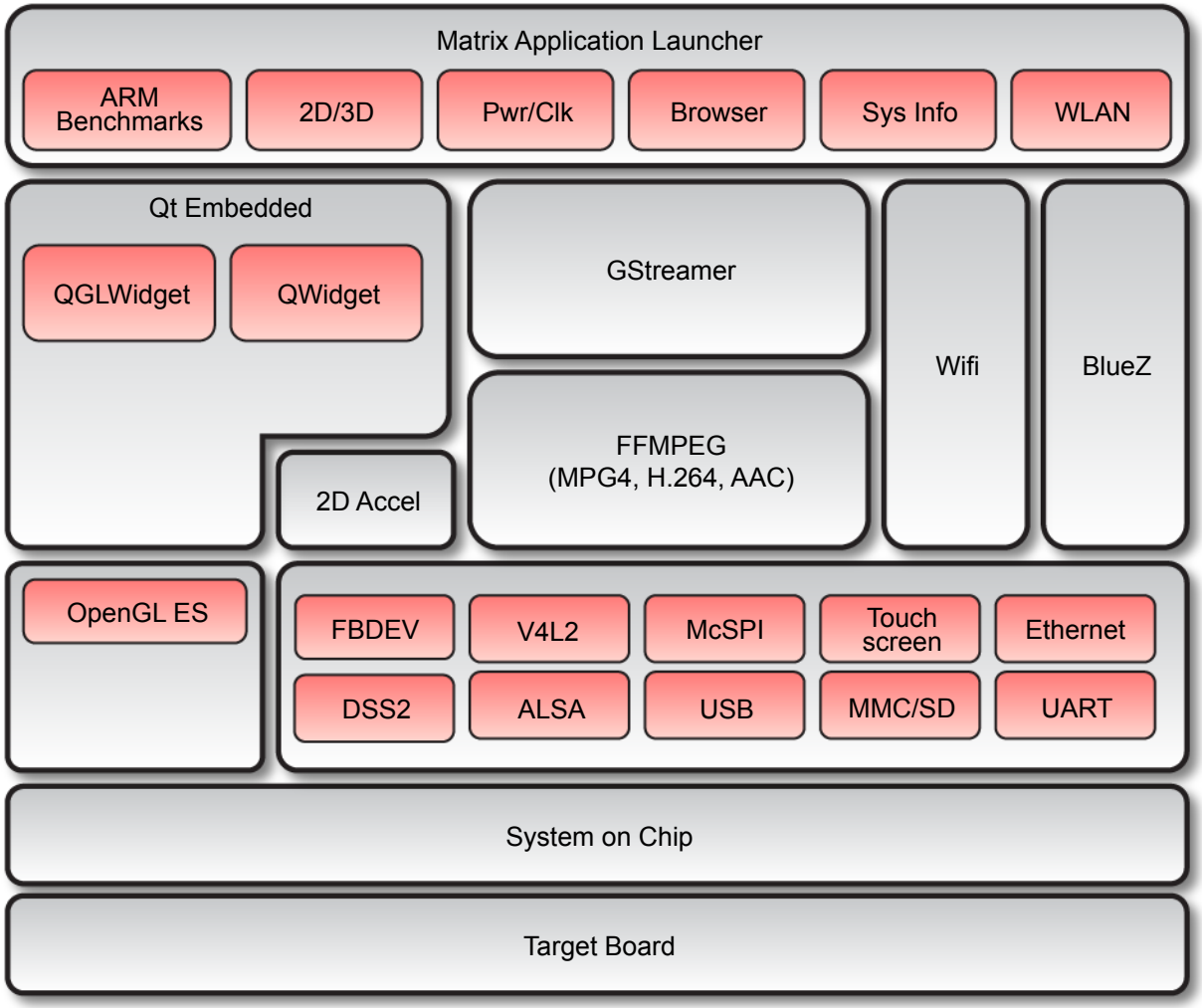
Conclusion

- Qt with QML is an excellent choice for developing highly performing GUIs on all sorts of affordable devices
- Android is growing in complexity/cost, but is an excellent choice if you need access to the App Market
- Tools for HTML5 have yet to emerge, but keep an eye out for them

Thank you!

Qt on TI Software Development Kits (SDK)

Software Components & Architecture



Backup stuff – matrix gui

ARM MPU – Sitara Microprocessors

Agenda

- Application Frameworks
- Qt/Webkit Overview
- 2D/3D Graphics
- Java
- Flash 10.x
- HTML5/CSS3
- DSS Features
- Examples
  - Matrix GUI
  - Matrix TUI

Qt Embedded / Webkit

- Qt is cross-platform application and UI framework.
- Qt provides a well defined API that can make development quick and easy.
- Webkit
  - Well accepted open source web browser
  - Rapidly create real-time web content and services
  - Use HTML and Java Script integrated in native code
- 3D Graphics with OpenGL and OpenGL ES
  - Easily incorporate 3D Graphics in your applications
  - Get maximum graphics performance
- Multithreading support
- Network connectivity
- Advanced GUI development