Linux Qt Graphical User Interface (GUI) Development

In this session, we will cover Qt GUI development tools including: Qt Creator for remote debug and Qt Designer for designing a UI. You will create your own Terminal GUI from scratch, add custom skins, implement a soft keyboard and examine a more complex example.

LAB:  
July 2012
Creative Commons Attribution-ShareAlike 3.0 (CC BY-SA 3.0)

You are free:
- to Share — to copy, distribute and transmit the work
- to Remix — to adapt the work
- to make commercial use of the work

Under the following conditions:

1. Attribution — You must give the original author(s) credit

2. Share Alike — If you alter, transform, or build upon this work, you may distribute the resulting work only under a license identical to this one.

With the understanding that:

- Waiver — Any of the above conditions can be waived if you get permission from the copyright holder.
- Public Domain — Where the work or any of its elements is in the public domain under applicable law, that status is in no way affected by the license.
- Other Rights — In no way are any of the following rights affected by the license:
- Notice — For any reuse or distribution, you must make clear to others the license terms of this work. The best way to do this is with a link to this web page.

CC BY-SA 3.0 License:
http://creativecommons.org/licenses/by-sa/3.0/us/legalcode
Pre-work Check List

- Installed and configured VMWare Player v4 or later
- Installed Ubuntu 10.04
- Installed the latest Sitara Linux SDK and CCSv5
- Within the Sitara Linux SDK, ran the setup.sh (to install required host packages)
- Using a Sitara EVM, followed the QSG to connect ethernet, serial cables, SD card and 5V power
- Booted the EVM and noticed the Matrix GUI application launcher on the LCD
- Pulled the ipaddr of your EVM and ran remote Matrix using a web browser
- Brought the USB to Serial cable you confirmed on your setup (preferable)
Agenda

• Introduction to QT

• Hands on Sessions
  – Manual helloworld
  – Using QT Creator
  – Making a GUI
  – Enhancing a GUI
  – Exploring existing examples and demos
    • Using an SGX accelerated Demo

Questions
**QT Framework**

» 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
Sitara SDK and QT

• QT Embedded is a component of the Sitara SDK
  – Includes libraries for building and running
  – Contains demo and example applications
  – Pulled in to the SDK via the Arago Framework.

• QT Creator is the GUI build tool
  – Must be downloaded in installed in your Host Linux development machine.
  – Requires some set up.
  – Can be used to rapidly develop GUI applications
Sitara Linux Stack

Matrix Application Launcher

- ARM Benchmarks
- 3D Graphics
- USB Profiler
- Power Manager
- Web Browser
- System Information
- WLAN GUI

Qt Embedded

- QGLWidget
- QWidget

Gstreamer/FFMPEG

- Wifi
- BlueZ

MPEG4, H.264, MPEG2, AAC, ...

OpenL3 ES

- FBDEV
- V4L2
- ALSA
- VISA
- McSPI
- Touch screen
- Ethernet
- UART

DSS2
- I2C
- USB
- McBSP
- MMC/SD

Sitara ARM Processor

Target Board
Creating Your First GUI – Hello World!

• Install the Sitara SDK on you Host 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

```cpp
#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.
Qt Framework – Application Classes

Core

GUI

OpenGL®, WebKit
Scripting, Multimedia
Networking, XML
Database, Unit Testing
2D Graphics Canvas, Declarative

Multithreading
Start Lab 1 here: Hello World Command Line

• Lab 1:
  – Get familiarized with where QT components are located in the Sitara SDK
  – Get familiarized with the gcc build tools in the SDK
  – Get familiarized with the environment-setup script
  – Create a helloworld GUI
  – Manually build and run it on the target.
QT Creator – Development tools

- Downloading and setting up:
Start Lab 2 here: QT Creator Hello World

• Lab 2:
  – QT Creator
    • Setup QT Creator within the Sitara SDK.
      – Specify a toolchain for cross compiling
      – Communicate with the target
        » Copy executables over and run them
        » Debug them
      – Easily add a helloworld label, run and execute.

• If time, investigate how much the QT helloworld uses cpu and memory.
QT Creator Help

• Selecting Help from menu bar brings up the QT Creator Manual

• There is extensive documentation at http://doc.trolltech.com/
  • Select the version of your QT embedded.

• Example - Search for Qwidget Class Reference
  • You get a description of the class
  • You get all the classes which inherit Qwidget (Most Widgets)
  • You get all the class members including:
    • properties, public functions, public slots, …..
Widgets

• 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 …
    
    ```cpp
    QPushButton * myButton = new QPushButton(...);
    myButton->doSomethingAPI();
    ```
  – Refer to online documentation at
    – Tip – Documentation is arranged using class names.
Start Lab 3 here: Terminal Project

• Lab 3:
  – Creating a GUI using QT Creator
    • Add widgets to the User Interface (ui)
    • Use Layouts to organize the widgets
    • Use Signals and Slots to make the Push Buttons do something
Signals & Slots

• Signals & Slots
  – Signals and slots are used for communication between objects. The signals and slots mechanism is a central feature of Qt and probably the part that differs most from the features provided by other frameworks.

• Signals
  – Events occur and cause a signal
    • Widgets contain a list of predefined signals, but you can subclass a widget to add your own signal
    • Example – button press, or Process complete
  – Slots are the functions which are assigned to handle a signal.
    • Widgets contain a list of predefined slots.
    • You can subclass a widget and add your own slots.

• Example – we will see a good example of signal and slots during LAB 4. when we debug the soft keyboard widget.
Start Lab 4 here:
Enhancing the Terminal project

• Lab 4:
  – Enhancements
    • Add a webview
    • Make the GUI fullscreen
    • Add a Soft Keyboard
    • Add style sheets
    • If time, debug keyboard.cpp to see how signals and slots work.
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 T.I.
  – 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

• Use QT Creator to pull in the project and build and run it on the target.

• 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.
Start Lab 5 here:
Exploring Existing Demos and Examples

• Lab 5:
  – See how many other demo and examples are already in the SDK
  – Run a SGX accelerated demo
  – If time, pull in an existing demo and build and run it.
Additional Links

• Recent Field Webex Training sessions
  – vmare image contains all the information needed for that presentation

• More on the Keyboard Template:

• How to add SGX acceleration and much more

• Qt Creator setup wiki:
Thank you!
Back up Slides
Introduction to Qt
QT Demo – Application Example Projects

Use a quick Camtasia video to point out the demos that exist and how easy it is to pull one in and start developing.
What’s Qt?

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

<table>
<thead>
<tr>
<th>Desktop OS</th>
<th>Embedded OS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows</td>
<td>Embedded Linux</td>
</tr>
<tr>
<td>Linux/X11</td>
<td>Symbian</td>
</tr>
<tr>
<td>Mac OS</td>
<td>Meego / Maemo</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Year</th>
<th>Events</th>
</tr>
</thead>
</table>
| 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

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

<table>
<thead>
<tr>
<th>Qt Release</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Qt SDK for Windows</td>
<td><a href="http://get.qt.nokia.com/qtsdk/qt-sdk-win-opensource-2010.05.exe">http://get.qt.nokia.com/qtsdk/qt-sdk-win-opensource-2010.05.exe</a></td>
</tr>
<tr>
<td>Qt SDK for Linux</td>
<td><a href="http://get.qt.nokia.com/qtsdk/qt-sdk-linux-x86-opensource-2010.05.1.bin">http://get.qt.nokia.com/qtsdk/qt-sdk-linux-x86-opensource-2010.05.1.bin</a></td>
</tr>
<tr>
<td>Qt Framework for Embedded Linux</td>
<td><a href="http://get.qt.nokia.com/qt/source/qt-everywhere-opensource-src-4.7.2.tar.gz">http://get.qt.nokia.com/qt/source/qt-everywhere-opensource-src-4.7.2.tar.gz</a></td>
</tr>
</tbody>
</table>

- 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

1. Build Qt for target
2. Design the UI in Qt designer
3. Add necessary event handlers
4. Add necessary application code
5. Create .pro file
6. Build & Install
Qt Graphics Operations

• Framework operations are of 2 types
  – (1) Widget surface creation
  – (2) Composition and blit to display

• Widget surfaces can be created from 2D or 3D operations
  – Ex, a 3D-ultrasound scan model on a window, is created by an openGL application, but still rendered to a 2D surface

• Composition and blit to display are 2D
  – (Most displays are 2D)
Painting in Qt

- **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
    • QGL PixelBuffer
    • 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

Qt Application

Qt for Embedded
(QPainter, QPaintEngine)

EGL
OpenGL ES | OpenVG

DirectFB
(Linux only)

Framebuffer

Operating System Kernel
(e.g. Embedded Linux, Windows CE)
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
Qt on TI Software Development Kits (SDK)
Software Components & Architecture

Matrix Application Launcher
- ARM Benchmarks
- 2D/3D
- Pwr/Clk
- Browser
- Sys Info
- WLAN

Qt Embedded
- QGLWidget
- QWidget

GStreamer

Wifi

BlueZ

FFMPEG
(MPG4, H.264, AAC)

OpenGL ES

FBDEV
V4L2
McSPI
Touch screen
Ethernet

DSS2
ALSA
USB
MMC/SD
UART

System on Chip

Target Board
Example Applications

- Matrix GUI Application Launcher provided in the SDK
  - Built with QT utilizing Webkit.
How to get Started
QT Creator – Development tools

Address Book Example

Tags: tutorial part1, tutorial, addressbook

This first part covers the design of the basic graphics user interface GUI for our address book application.

Building and Running an Example...

Tags: qt creator, build, compile

You can test that your installation is successful by opening an existing example application project.

Creating a QT Quick Application...

Tags: qt quick, qml, components, symian, visual designer, qt creator

This tutorial describes how to use QT Creator to create a simple Qt application, showing the System Information Mobility API to fetch battery information from the device. The user interface for the application is designed using Qt Quick.

Creating a Qt Widget Based...

Tags: qt, c++, text, qt designer, qt creator

This tutorial describes how to use QT Creator to create a small Qt application, Text Finder.

Creating a Qt Widget Based Mobile...

Tags: qt, c++, mobile, qt mobility, qt creator

This tutorial describes how to use QT Creator to create a small Qt application that uses the System Information Mobility API to fetch battery information from the device. The user interface for the application is designed using Qt widgets.
QT Demo – Application Example Projects

Use a quick Camtasia video to point out the demos that exist and how easy it is to pull one in and start developing.
About QT

• QT Embedded version 4.7.2
  – Cross Platform
  – Opensource can download and build yourself here: http://doc.qt.nokia.com/4.7/qt-embedded-linux.html

• QT Embedded SDK has been integrated into the Sitara SDK
  – Can be built using OE
  – Can specify preferred version of QT.

• License – LGPL version 2.1.

• Tools
  – QT Creator latest version 2.2.1 for MAC/Linux/Windows
References

• SWApps Sharepoint – Sitara Bootcamp 2011
Backup stuff – matrix gui
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 a 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 JavaScript 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
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 a 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

/usr/bin/app1
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**

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

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

/* This section 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

```html
<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 mandatory
• appParameters and any other fields are optional
Matrix GUI – HTML Header

```html
<body>
  <div class="topBar">
    <object type="image/svg+xml"
      data="/usr/share/matrix/images/tex.svg" >
      <img src="/usr/share/matrix/images/tex.svg" />
    </object>
    <div id="header">Matrix Application Launcher p1</div>
  </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.
Matrix Text User Interface (TUI)

» Text User Interface for the serial console

» For platforms with no LCD

» For performance measurements without Qt/Webkit overhead

» Same functionality and hierarchy as Matrix GUI
Matrix Text User Interface (TUI)

» Text User Interface for the serial console

» For platforms with no LCD

» For performance measurements without Qt/Webkit overhead

» Same functionality and hierarchy as Matrix GUI
Matrix Text User Interface (TUI)

» Text User Interface for the serial console

» For platforms with no LCD

» For performance measurements without Qt/Webkit overhead

» Same functionality and hierarchy as Matrix GUI