Agenda

1 Qt for Embedded Overview
2 The Fun of Cross-Compiling
Architecture overview

Qt Application
- C++
- Java™
- others...

Qt Libraries
- Core
- GUI
- Database
- XML
- Scripting
- Multimedia
- Network
- Font Engine
- OpenGL
- WebKit

Operating System
- Windows
- Mac
- Linux
- Windows CE
- E Linux
Which OS is special from a UI perspective?

Operating System

Windows  Mac  Linux  Windows CE  E Linux
Something missing?

Native UI

- Win32 UI
- Carbon / Cocoa
- X11
- CE libs
- Windows
- Mac
- Linux
- Windows CE
- E Linux
QWS to the rescue

Native UI

Win32 UI  Carbon / Cocoa  X11  CE libs  QWS

Windows  Mac  Linux  Windows CE  E Linux
What is QWS?

- Qt Windowing System
  - Lightweight UI Server
    - window manager
    - plug-able font engines (qpf, freetype, Agfa Monotype, Arphic Font Engine, ...)
  - Connects Qt with low-level devices
    - Keyboard / Keypad
    - Mouse / Touchscreen
    - Display
QWS - Gfx Drivers

- LinuxFb
- SVGAlib
- directFB
- VNC
- Virtual Frame-buffer

- /dev/fb
- VGA cards
- HW
- Network
- Local Socket
QWS

Demo
QWS - Painting

Paint Event

Widget

Backing Store

Expose Region

Server

Screen Driver

Window Surface

Window Surface
Why (not) QWS?

- Lightweight implementation + legacy free
  - Multiple screens
  - Rotated screens
  - Top-level transparency & composition
  - Network transparency through VNC protocol
  - Both in-process or client-server mode

- Tailored for Qt - coexistence with other toolkits tricky
  - J2ME successfully integrated

- Smaller ecosystem
Fine-tune library size with QConfig

<table>
<thead>
<tr>
<th>Id</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TOOLBAR</td>
</tr>
<tr>
<td></td>
<td>MENUBAR</td>
</tr>
<tr>
<td></td>
<td>WORKSPACE</td>
</tr>
<tr>
<td></td>
<td>PROGRESSBAR</td>
</tr>
<tr>
<td></td>
<td>RESIZEHANDLER</td>
</tr>
<tr>
<td></td>
<td>RUBBERBAND</td>
</tr>
<tr>
<td></td>
<td>SPLITTER</td>
</tr>
<tr>
<td></td>
<td>SIGNALMAPPER</td>
</tr>
<tr>
<td></td>
<td>SIZEGRIP</td>
</tr>
<tr>
<td></td>
<td>SLIDER</td>
</tr>
<tr>
<td></td>
<td>DIAL</td>
</tr>
<tr>
<td></td>
<td>SCROLLBAR</td>
</tr>
<tr>
<td></td>
<td>SCROLLAREA</td>
</tr>
<tr>
<td></td>
<td>GRAPHCISVIEW</td>
</tr>
<tr>
<td></td>
<td>PRINTPREVIEW...</td>
</tr>
<tr>
<td></td>
<td>MDIAREA</td>
</tr>
<tr>
<td></td>
<td>TEXTEDIT</td>
</tr>
<tr>
<td></td>
<td>SYNTAXHIGHLIGHTER</td>
</tr>
<tr>
<td></td>
<td>TEXTBROWSER</td>
</tr>
<tr>
<td></td>
<td>SPINWIDGET</td>
</tr>
<tr>
<td></td>
<td>SPLASHSCREEN</td>
</tr>
<tr>
<td></td>
<td>STACKEDWIDGET</td>
</tr>
<tr>
<td></td>
<td>TABWIDGET</td>
</tr>
<tr>
<td></td>
<td>STATUSBAR</td>
</tr>
<tr>
<td></td>
<td>STATUSTIP</td>
</tr>
<tr>
<td></td>
<td>TABLEWIDGET</td>
</tr>
<tr>
<td></td>
<td>TOOLBUTTON</td>
</tr>
</tbody>
</table>

GRAPHCISVIEW

QGraphicsView

Supports the graphicsview classes.

Section:
- Widgets

Requires:
- [On] SCROLLAREA

Required for:
- [On] PRINTPREVIEWWIDGET
- [On] GRAPHICSSVGITEM
Agenda

1. Qt for Embedded Overview
2. The Fun of Cross-Compiling
Building Qt embedded for your host

- ./configure -embedded -qvfb
- make
- sudo make install
Building Qt embedded for your target

- ./configure -embedded arm -xplatform qws/linux-arm-g++
- make
- sudo make install
Two tool chains

- Host tool chain
  - Tools for building Qt (qmake, moc, uic, rcc...)
- Target tool chain (cross-compiler)
  - Everything that runs on the target (libs, apps...)
Cross Compiling

- `g++ main.cpp`
  - `a.out`: ELF 32-bit LSB executable, Intel 80386

- `arm-linux-g++ main.cpp`
  - `a.out`: ELF 32-bit LSB executable, ARM

- `ntomips-g++ main.cpp`
  - `a.out`: ELF 32-bit MSB executable, MIPS
Cross Compiling

- Cross compiler can compile & link
  - ... but apps can’t be executed (unless using scratchbox)
- Special care needed for configure tests
Workarounds - configure tests

```c
int main()
{
    printf("size of pointers: %d\n", sizeof(void*));
}
```
Workarounds - configure tests

```c
int main()
{
    STATIC_ASSERT(sizeof(void *) == 4);
}
```
Workarounds - configure tests

// "MostSignificantByteFirst"
short msb_bigendian[] = { 0x0000, 0x4d6f, 0x7374, 0x5369,
                        0x676e, 0x6966, 0x6963, 0x616e, 0x7442, 0x7974, 0x6546,
                        0x6972, 0x7374, 0x0000 };

// "LeastSignificantByteFirst"
short lsb_littleendian[] = { 0x0000, 0x654c, 0x7361, 0x5374,
                         0x6769, 0x696e, 0x6966, 0x6163, 0x746e, 0x7942, 0x6574,
                         0x6946, 0x7372, 0x0074, 0x0000 };
Thank You!

And happy embedded hacking!