Qt on Embedded Systems

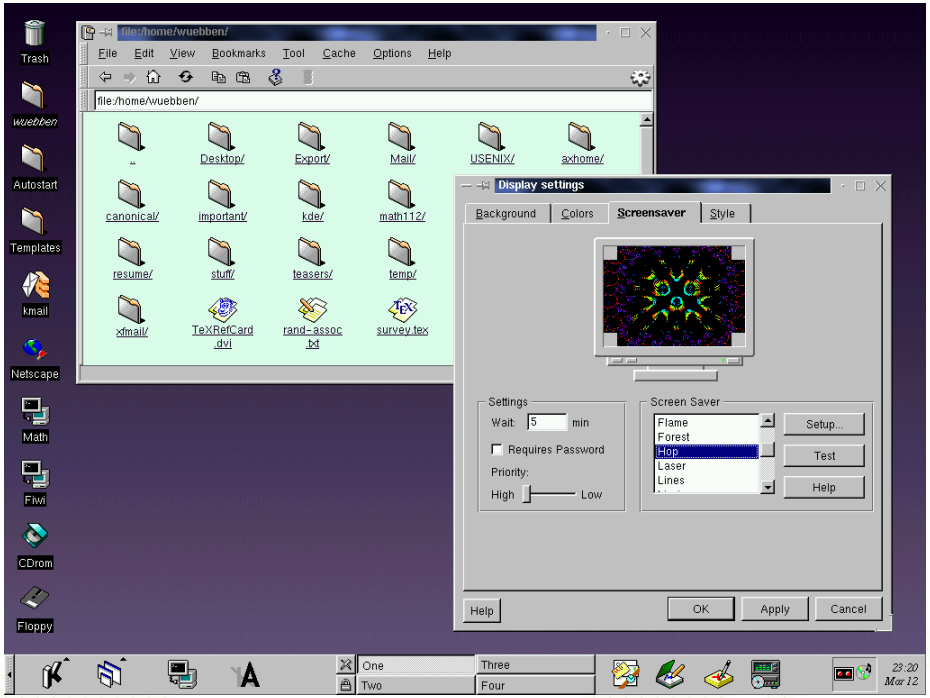
13. November 2012

Lars Knoll <lars.knoll@digia.com>

digia

The early years

- Started on Windows and X11
  - Used native apis
  - All painting done by the underlying Windowing system
  - Every widget a native window



digia

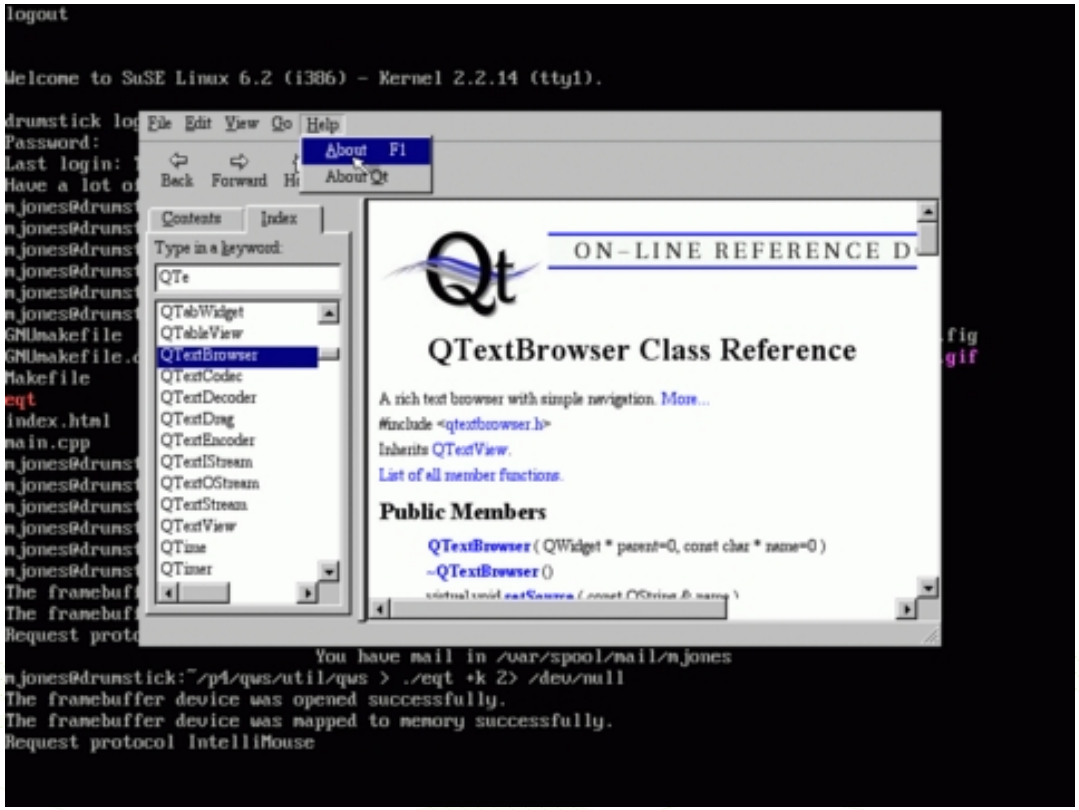
Embedded systems

- 1999:
  - 240x320 screens on high end embedded systems
  - 16MB RAM and ROM
  - Faster processors some HW acceleration for graphics
- Linux became an interesting option
  - No available UI solution, X11 not suited for embedded systems
  - Linux had a framebuffer
  - We had a prototype:
    - QImagePaintDevice
    - Draw 2d graphics into a raster buffer

digia

Qt Embedded and QWS

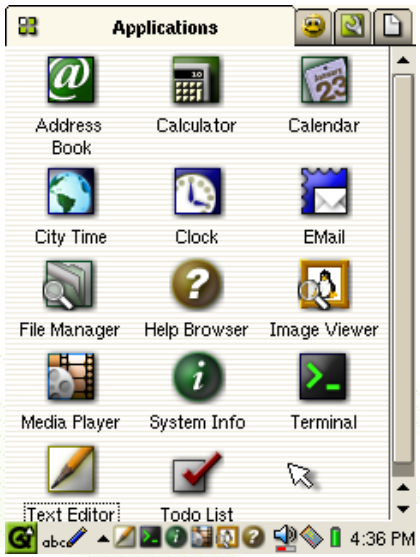
- Started development in 1999
  - Lean/simple stack
  - Run on 8MB (or less) RAM/ROM
- windowing system included into the framework
- Any app could be the server process
- Single and multi process modes
- Software drawing on Linux framebuffer out of the box
- Abstractions for limited graphics acceleration
- Minimal stack fit onto a 1.4'' floppy

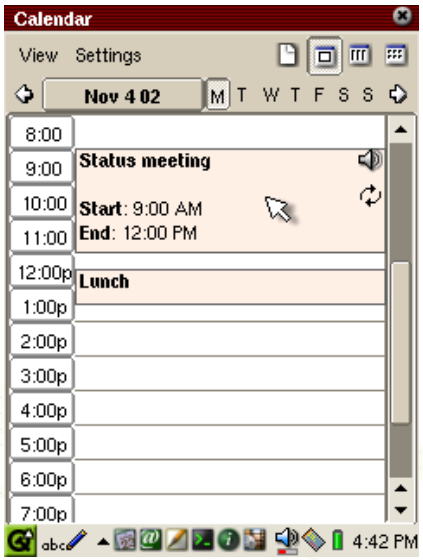


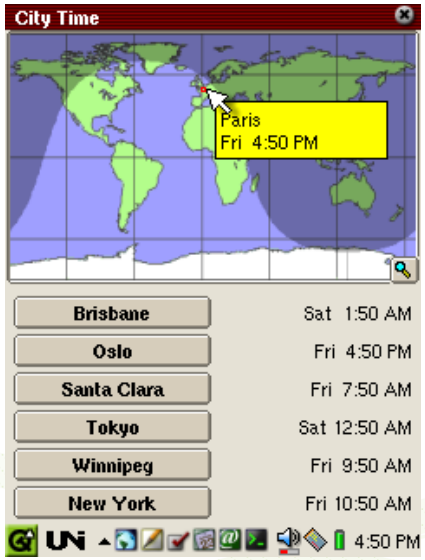
digia

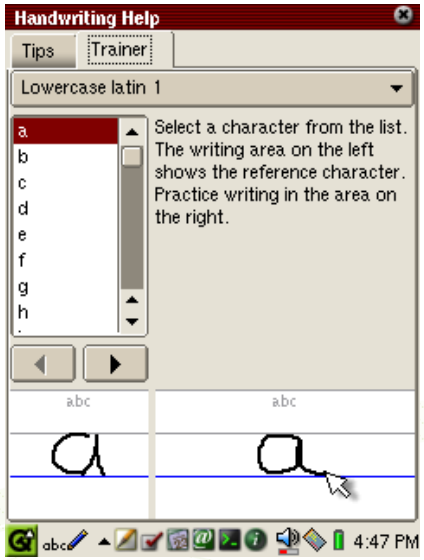
Qt Palmtop Environment

- Just the framework not enough
- Needed some demo apps: QPE
- Rebranded as Qtopia a little later









digia

Qtopia

- Sharp bought it for their Zaurus PDA in 2001
  - From demo to shipping in 6 months

digia

Industrial embedded

- Used all possible combinations
  - From minimal config
  - To all of Qtopia

digia

… Meanwhile in desktop land …

- 2002: Qt 3 shipped
  - Lots of new features for existing desktop customers
  - Too fat and slow for embedded devices
  - Mainly ignored by embedded customers
- 2005: Qt 4 shipped
  - Brought back most of the required speed for embedded devices
  - New functionality making it interesting again
  - Many Qt/e features migrated into the desktop versions
    - Alien widgets
    - Painting abstraction (QPaintEngine)

digia

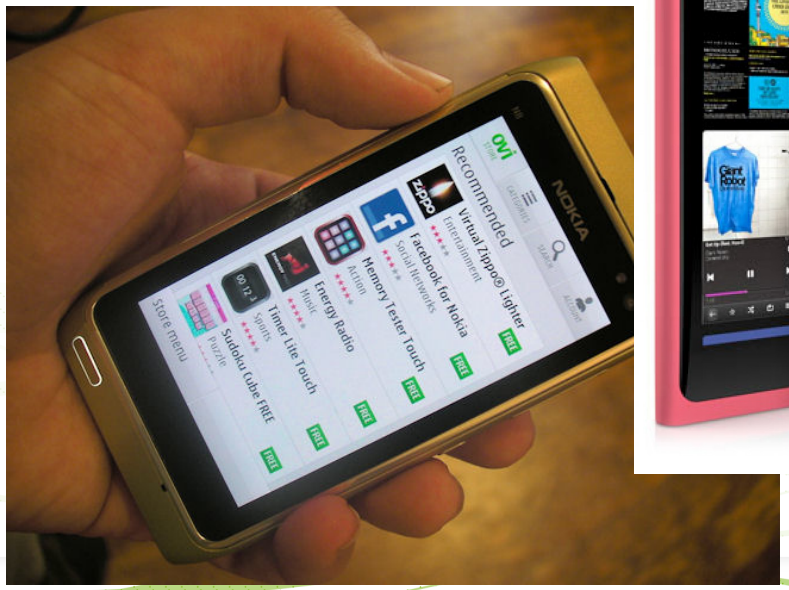
IP/VoIP phones, Greenphone

- Very demanding UX requirements
- Touch based
- Fluid animations
- Integrated Video

digia

Nokia

- 2008: Nokia bought Trolltech
  - Fully shifted focus from Desktop to Embedded
  - Performance, performance, performance
  - Symbian and Linux (Maemo/MeeGo)
- Huge added investment
  - Tooling
  - Mobility APIs
- **Qt went LGPL**



digia

Rethinking User interfaces

- VoIP and mobile phones showed limits of Qt's architecture
  - Widgets rectangular items
  - Animations almost impossible
  - No way to realize the UI designs in a clean way

→ **Two research projects**

- Kinetic project (Oslo)
  - QGraphicsView an existing scene graph
  - Animations, States and Transitions
  - Some added sugar on top of C++
- QML (Brisbane)
  - More radical approach
  - New XML based language
  - Do we need C++ APIs?
  - Maybe a different scene graph?

digia

Qt Quick (version 1)

- Take most of the ideas from Brisbane
  - Change from XML to own language (extension to Javascript)
  - Use QGraphicsView
- QML Language
  - Javascript based
  - Object trees
  - Declarative syntax
  - Property bindings
- Optimised for UI design
- Small C++ API
- Easy to bind to and extend from C++

```
Rectangle {
    width: 320
    height: 240;
    property color textColor: “black”
    Text {
        anchors.centeredIn: parent
        text: “Hello World!”
        color: parent.textColor
    }
}
```

digia

Rethinking window system integration

- Symbian port showed that our architecture was flawed
  - A new port of Qt extremely hard to do
  - Took too long
- QWS reaching it's limits
  - Perfect in the 90s with limited 2D acceleration
  - OpenGL/OpenVG support very hard to do
  - Porting to other OSes very challenging (VxWorks, QNX)
- HW adaptation very hard
  - Write complete port of Qt (50k LOC) or
  - Hack Qt/embedded (not much less work)

  → Both very error prone

digia

Project Lighthouse

- Qt Platform Abstraction (QPA)
  - Clean API to encapsulate the windowing system
- Released in 4.8
  - Android and iOS ports by 3rd parties prove the design
  - EGL full screen / OpenGL backend with ~2000 LOC
  - Great support for HW acceleration
- No own windowing system
  - Multi process through e.g. Wayland

digia

… Come Qt 5 …

- Completely based on QPA
- Qt Quick (v2) fully OpenGL (ES) based
  - OpenGL Scene graph
  - Separate rendering thread
  - Fluid 60FPS UIs
- Separate Qt Widgets and Qt Quick
  - Allow for a leaner stack on embedded devices

→ Ideas from embedded have entered mainline Qt and all ports

- Release timeline for 5.0
  - Beta 2 released 13. Nov.
  - Final in December

digia

QPA options for Linux

- DirectFB
  - Blitting acceleration
  - Input handling
  - First port contributed to Qt Project
  - OpenGL support available with some Vendor integration
  - ~ 3000 LOC
- XCB
  - X11 support
  - ~ 18.000 LOC
- Minimal and minimal-egl
  - As simple as possible, helps getting started with a custom plugin
- Experimental plugins
  - KMS, OpenWF, linuxfb
- **EGLFS & Wayland**

digia

EGLFS

- Full Screen, single surface
- EGL used for Surface creation
- OpenGL and SW rasterization for drawing
- Directly reads from input devices
- Device discovery through udev
- Single process only
- Very easy to integrate
- ~ 2000 LOC

→ **Great option for single process UIs if EGL and OpenGL is available**

digia

Wayland

- Qt Wayland module
- Works with Wayland 1.0
- Fully functional QPA plugin for Wayland
  - ~ 7000 LOC
  - Supports Clipboard, DnD, Touch input
- Qt Compositor API
  - Build your own wayland compositor
  - Makes it very simple to manage surfaces
  - Qt Quick integration, write your Compositor using QML
  - ~ 11.000 LOC
- Compatible with other wayland clients and servers

→ **Best solution for multi process environment, integrates with other frameworks**

digia

Rasberry/Pi

Demo

digia

HW without OpenGL

- No Qt Quick 2
- Mesa + LLVM Software OpenGL possible
  - Would allow for Qt Quick
  - LLVM untested on ARM (might work with LLVM > 3.1)

Possible QPA plugins:

- Wayland
  - shared memory buffers
- DirectFB
- Linuxfb
- Xcb (if you want X11)

digia

The future

- Wide variety of QPA backends existing today
  - Mac/Cocoa, Windows, QNX/BB10
  - Very easy to get started on a new HW or even new OS
  - The cross platform solution: Add **Android** and **iOS**
- Qt Quick components for Touch
  - Greatly simplifies UI creation
- Strong focus on embedded use cases & requirements
- High quality tooling support
  - Cross compiling, remote debugging
  - Easy deployment
  - Flashing

  → Integrated into Qt Creator

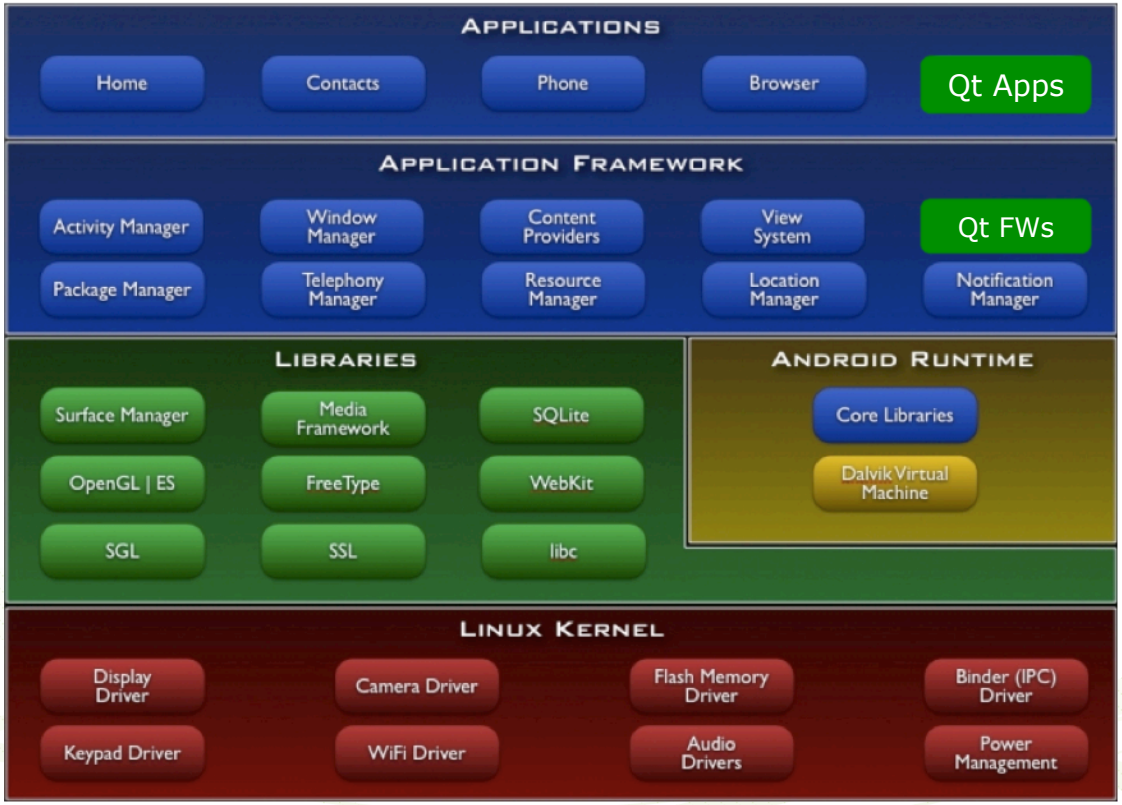
digia

Qt on Android

- Android port
  - Existing port on QPA for Qt 4.8: Necessitas
  - Bring to Qt 5
  - Fully integrate with existing Android stack
  - Offer a native runtime that keeps compatibility
  - Deployment solution
  - Bring Qt apps into the Android Marketplace
- Embedded on Android
  - Use Android base layer only
    - Kernel, Drivers, libc, OpenGL ES, Media Framework
  - Dalvik available, but not required (depending on use case)
  - Just starting the work, lots of open questions…

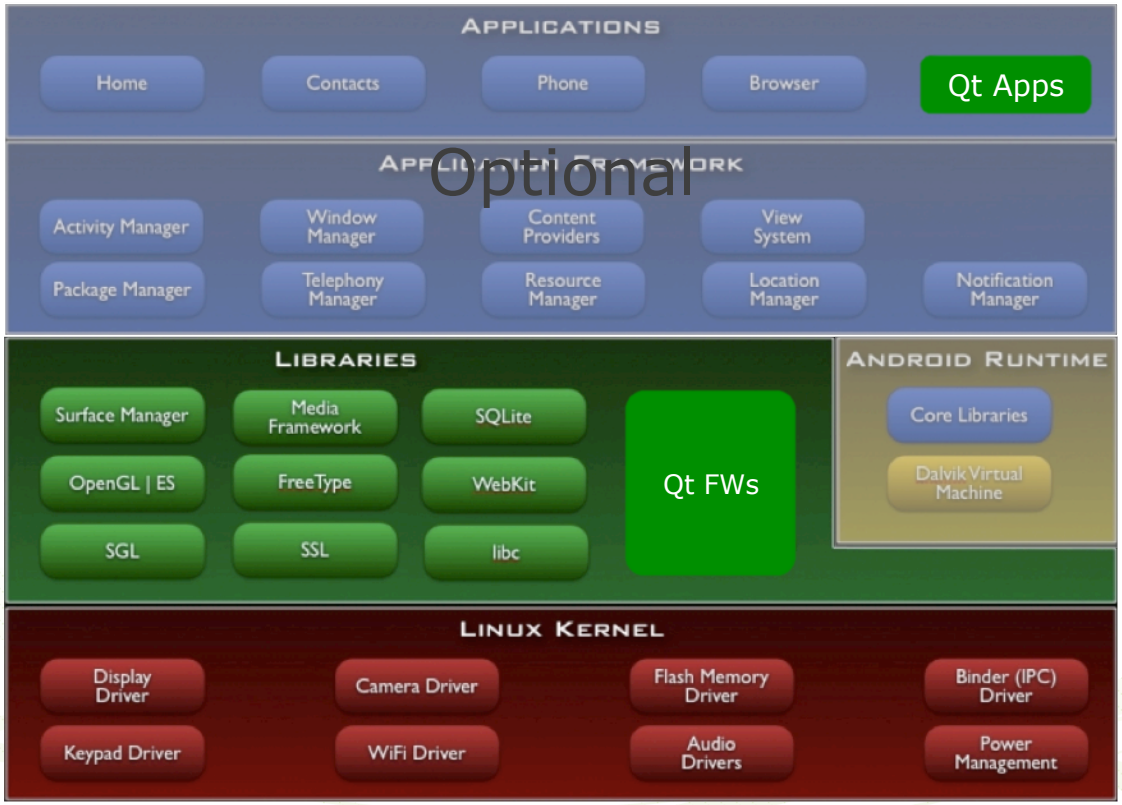
digia

Android port



digia

Embedded on Android



digia

Qt on Nexus

Demo

digia

Thank you!

digia