Qt Graphics on Embedded

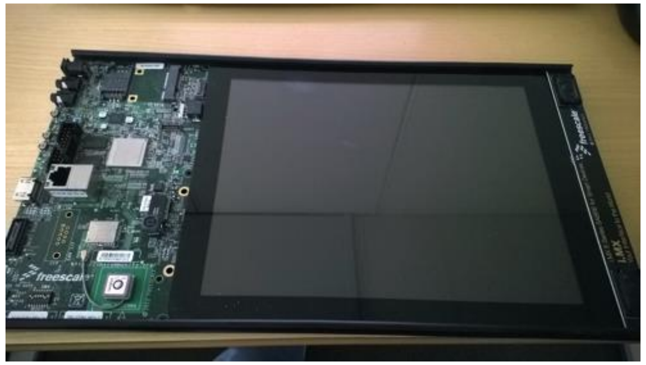
László Agócs
@alpqr
Senior Software Engineer, Maintainer for eglfs & friends
The Qt Company, Oslo, Norway

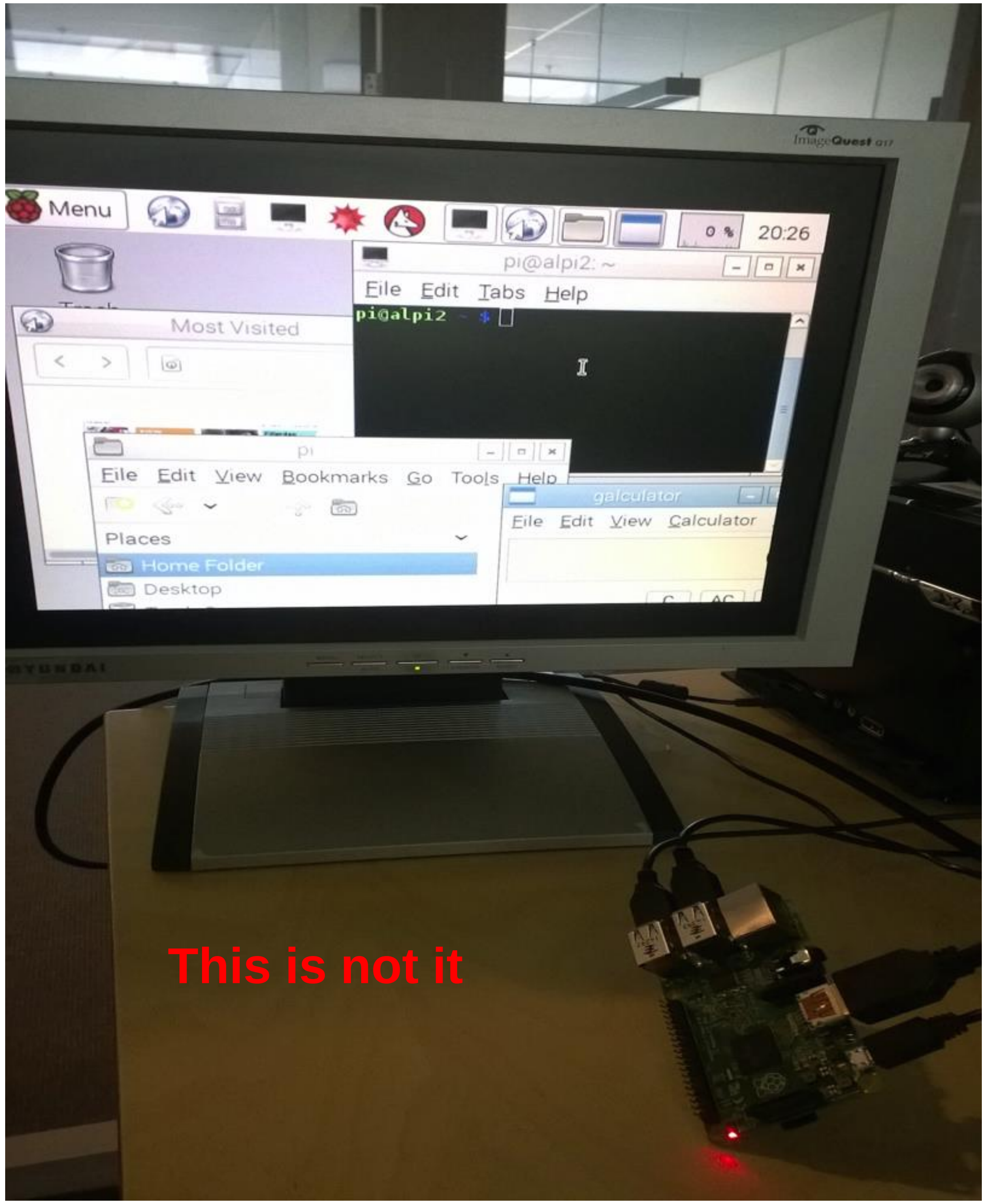
QtCon – Berlin - 3 September 2016

Agenda

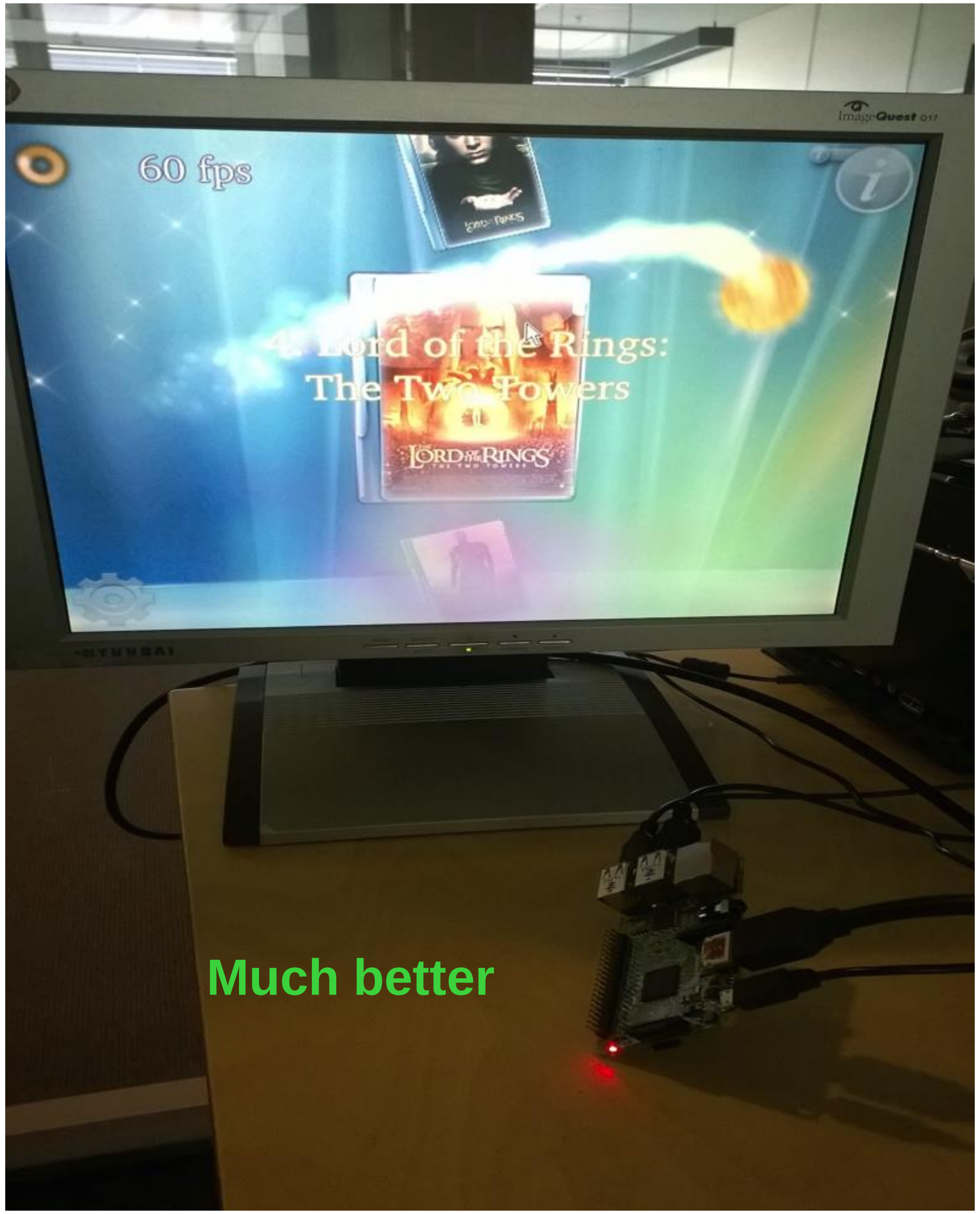
- Qt on Embedded Linux: building, docs
- Advancements in pure software rendering
- HW accelerated graphics, what's happening in eglfs
- What is (may be) on the table for the future
  - OpenGL optimizations, Path rendering, OpenVG, 2D acceleration, Emulator, Vulkan, CI, …

Embedded Development Boards





This is not it



Much better

Windowing Systems in Device Creation

- Run a Qt app in fullscreen as the only GUI app
  - eglfs. EGL + OpenGL ES
  - linuxfb. Pure software
  - directfb. Some 2D acceleration
- Wayland (QtCompositor (on top of eglfs) or Weston + platform plugin from QtWayland)
  - EGL + OpenGL (ES)
- X11 – **not recommended**
  - xcb. EGL + OpenGL ES or GLX + OpenGL

Building Qt on Embedded

- Building
    - Cross-compilation
        - Manual
        - Yocto
            - http://blog.qt.io/blog/2016/07/01/aligning-with-the-yocto-project/
        - Others
    - On target

Documentation

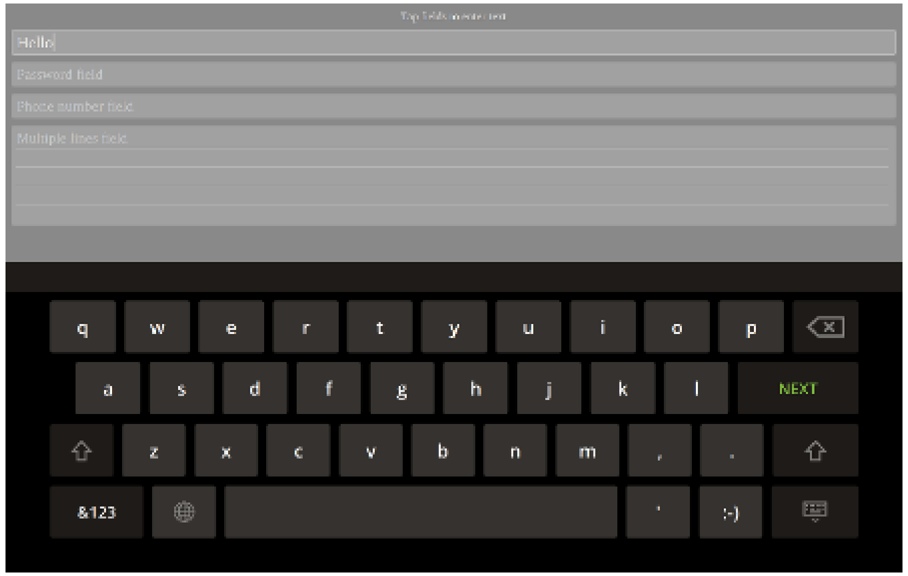
- https://doc-snapshots.qt.io/qt5-dev/embedded-linux.html
    - Bit hard to find: Supported Platforms → Embedded Linux
    - Enhanced in every version.
    - Covers manual building and graphics + input stuff. No Yocto or other integration specifics.
    - Can feel convoluted due to covering multiple overlapping approaches plus some legacy stuff.
- http://doc.qt.io/QtForDeviceCreation/index.html
    - This is for the **commercial** additions, including the pre-built reference images and SDKs, Windows host support, some extra tooling, etc.
    - Under the hood it is still eglfs and co.

GPU-less rendering

- Qt 5.8 brings the Qt Quick 2D Renderer into Qt Quick core
  - LGPLv3 + commercial, like the rest of qtdeclarative
- Partial update support
  - Dirty rect tracking, no more fullscreen updates, great news for linuxfb
- Qt Quick now builds when OpenGL is disabled due to -no-opengl or autodetection
- Qt WebEngine is functional too on top
- i.MX7 and similar will love this

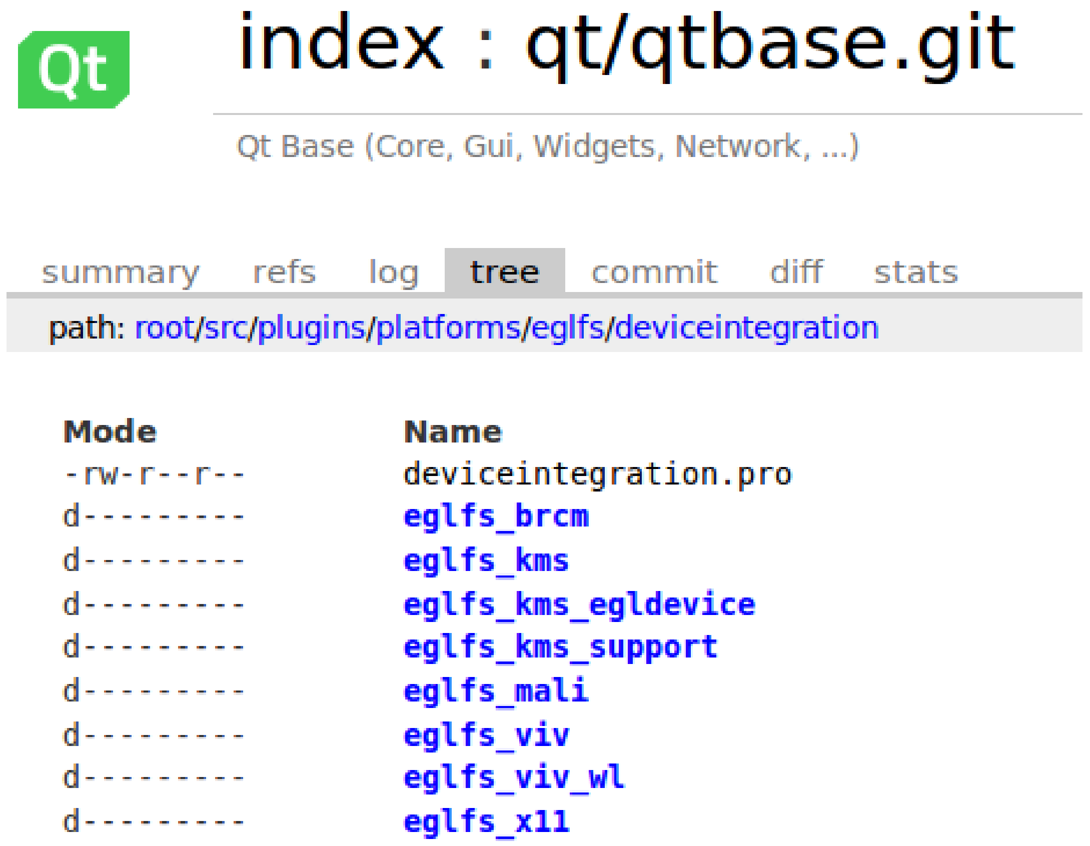
Speaking of license changes...

- Qt 5.7 brought some previously commercial components to open-source
  - GPLv3 + Commercial
    - Charts, Data Visualization, **Virtual Keyboard**



eglfs backends

- Development focus is on KMS/DRM
- Two variants: GBM (Mesa and others), EGLStream (NVIDIA)
- No big changes expected for others (Vivante fbdev, Mali fbdev, Broadcom dispmanx)



path: root/src/plugins/platforms/eglfs/deviceintegration

Mode	Name
-rw-r--r--	deviceintegration.pro
d---------	eglfs_brcm
d---------	eglfs_kms
d---------	eglfs_kms_egldevice
d---------	eglfs_kms_support
d---------	eglfs_mali
d---------	eglfs_viv
d---------	eglfs_viv_wl
d---------	eglfs_x11

KMS/DRM improvements

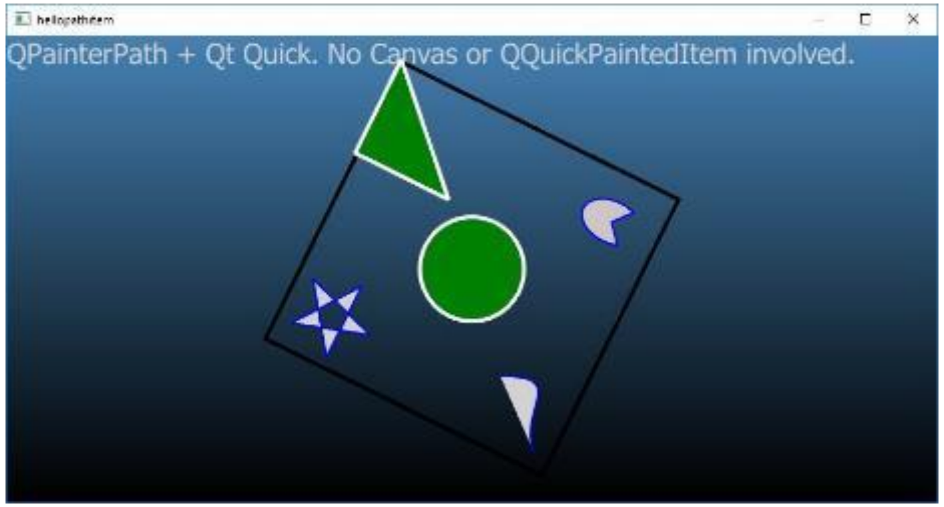
- EGLDevice/EGLOutput/EGLStream support since Qt 5.6
- Qt 5.7 unifies a lot, code sharing between the two, basic multi-display for EGLStream as well.
- Adds NVIDIA DRIVE CX (AArch64) device spec.
- Qt 5.8 enhances multi-display for EGLStream:
  - should now be on-par with GBM
  - virtual desktop mouse cursor
  - touchscreen config (which screen does it belong to?)

KMS/DRM improvements

https://doc-snapshots.qt.io/qt5-dev/embedded-linux.html#display-output

Future work plans. Qt 5.9 and beyond.

- OpenGL optimizations
  - Compressed textures
  - Pre-compiled shaders
  - Optimizations using ES 3.1 & 3.2
  - **Path rendering in Qt Quick**
    - Generic (triangulate, for OpenGL and D3D)
    - Software (something QPainter-based for the software backend)
    - GL_NV_path_rendering
      - vendor-specific, but available on desktop/mobile/embedded(!)



QPainterPath + Qt Quick. No Canvas or QQuickPaintedItem involved.

Future work plans. Qt 5.9 and beyond.

- 2D acceleration?
- Directfb platform plugin should still work but its future is somewhat unclear.
- OpenVG is kind of dead, but not fully. Prepare for potential surprises.
- Vendor-specific APIs (gal2d, g2d, …)
  - Perhaps.

Future work plans. Qt 5.9 and beyond.

- Everyone's favorite topic: Vulkan!
- Without wide-spread embedded adoption there's little motivation.
- Qt Quick gains an experimental D3D12 backend in Qt 5.8.
  - New low-level APIs present a lot of work and little reward for typical UIs.
  - There is a lot of room for research. (threads, threads, threads. but can be difficult to fit into an existing architecture.)

(Jetson/L4T does not count)

- So far encountered one *device creation oriented* embedded board with Vulkan support, but without a WSI. Some success with using GL_NV_draw_vulkan_image to integrate Vulkan rendering into Qt Quick apps.

Future work plans. Qt 5.9 and beyond.

- There are some ideas for basic enablers for setting up WSI and rendering. Maybe Qt 5.9, but likely later.
    - To enable cross-platform Vulkan into a QWindow or a window embedded into widgets.
- In the meantime: https://github.com/alpqr/qtvulkan
- Qt Quick backend for Vulkan: do not hold your breath. However, some people expressed interest in working on this, so who knows.

Future work plans. Qt 5.9 and beyond.

- Continous Integration?
- The Qt Project CI system has one configuration to do 32-bit ARM cross-compilation.
  - Only compilation, no tests run.
  - Does not check if the results are functional, e.g. graphics-wise.
- 64-bit ARM Linux (since Qt 5.7) was challenging. Some manual testing only. Then got regressions and JIT problems discovered along the way.
- Some work on-going to put Jetson TX1s into the test farm.
  - Lancelot already running (non-blocking graphics testing, Qt built on device, X11, not really device creation style but at least 64-bit ARM)

Future work plans. Qt 5.9 and beyond.

- Emulator?
- Part of Qt for Device Creation and the Qt Automotive Suite.
- Based on VirtualBox and streaming EGL/GLES commands between the VM running a Yocto-generated Linux image and the host.
- Plans for bringing multi-display and multi-process (Wayland) simulation capabilities to it.
    - New eglfs_emu backend.
    - QtWayland integrations for GL streaming.

Thank You!

www.qt.io