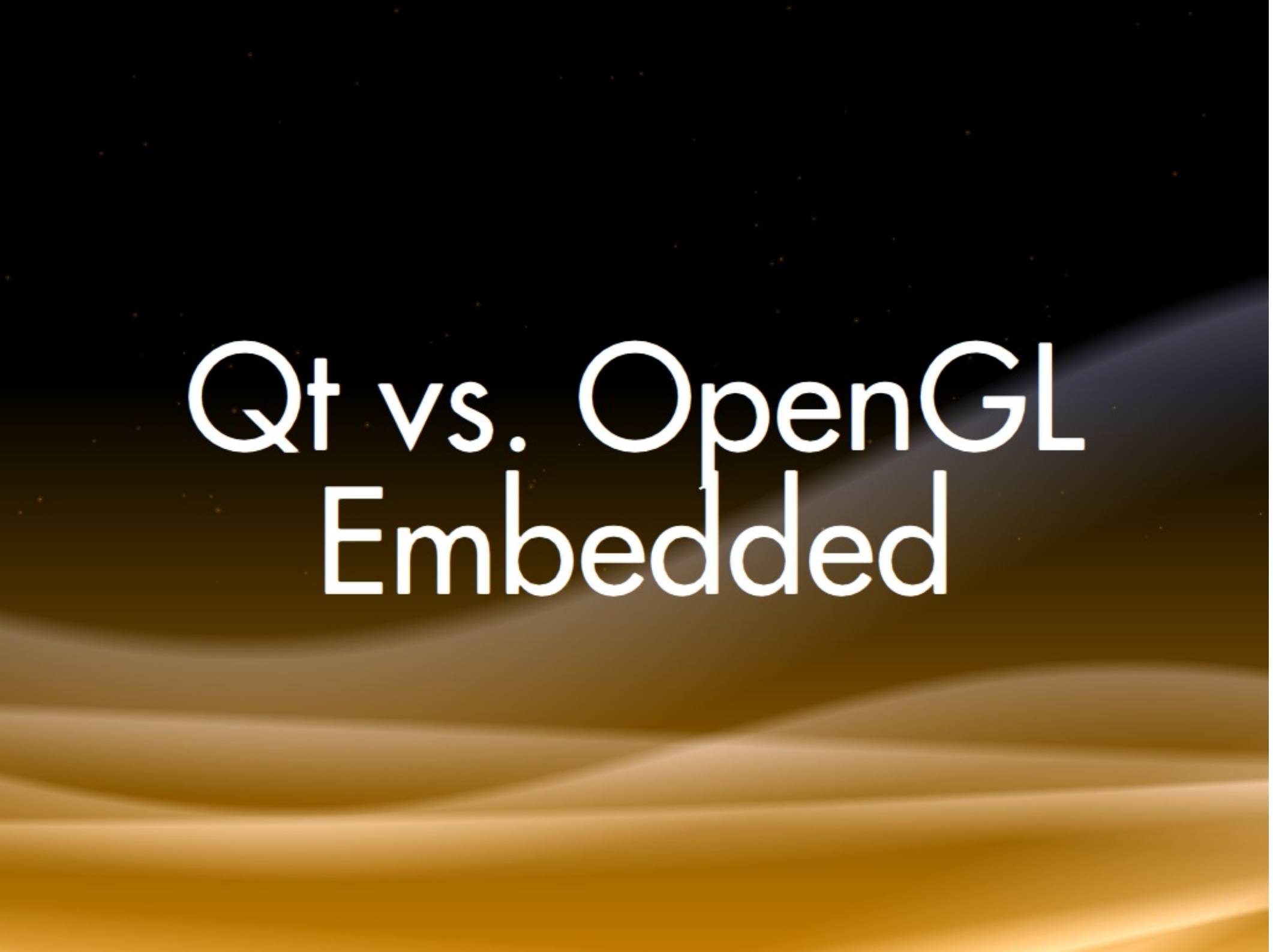

Qt vs. OpenGL
Embedded

Me

- Thomas Senyk
- Graduated in computer science in 2009
- Qt Professional Services Engineer at Nokia from 2009 to 2012
  - Helping Qt customers with embedded systems
- Since mid 2012 at Pelagicore
  - Focus on Qt in Automotive

... going out, beer, traveling, video games

Outline

- Glossary
- OpenGL on Embedded
  - Implementation
  - Platform integration - EGL
  - Windowing system
- Appendix

Perspective and Scope

- perspective: OpenGL ES 2.0 and Qt5 on embedded Linux
- scope: OpenGL (ES) 2.0+ with and without Qt5
- out of scope: Qt or OpenGL implementation details

Glossary

OpenGL

"OpenGL (Open Graphics Library) is a cross-language, multi-platform Application programming interface (API) for rendering 2D and 3D computer graphics."

[http://en.wikipedia.org/wiki/OpenGL]

OpenGL context

- An OpenGL context represents many things
  - all of the state
  - default framebuffer
- Think of a context as an object that holds all of OpenGL
- making a context current:
  loading and activating an OpenGL-instance
- OpenGL has NO API for context!
  - EGL (or another platform integration) is needed

[http://www.opengl.org/wiki/OpenGL_Context]

OpenGL on Embedded

OpenGL on Embedded

- Driver
- Platform integration
- Windowing system

Driver

- OpenGL ES2.0 API implementation
  - e.g. libGLESv2.so
- Platform integration
  - e.g. libEGL.so
- Probably kernel modules
  - e.g. galcore
- Maybe some helper libraries
  - e.g. libGLSLC.so, libGAL.so,…

Where do I get my drivers?

- Mesa
  - open source
  - support for some(!) graphics cards
  - CPU-based fall back implementation
  - not very common in the embedded segment!
- Chipset vendor
  - likely closed source / proprietary
  - hopefully highly optimized
  - ... might include kernel tree/patch
  - examples!!
- Angle
  - OpenGL ES 2.0 implementation on top of DirectX

Platform integration

OpenGL on platforms

- OpenGL API is platform agnostic!
  - no display
  - no window
  - no surface
  - not even a context!

  - no use-input, no event handling, ...
    ==> Qt

OpenGL on platforms

?

Platform

x11 / directFB / linuxFB / wayland

OpenGL on EGL

Platform

x11 / directFB / linuxFB / wayland

EGL

"EGL™ is an interface between Khronos rendering APIs such as OpenGL ES or OpenVG and the underlying native platform window system.

It handles graphics context management, surface/buffer binding, and rendering synchronization and enables high-performance, accelerated, mixed-mode 2D and 3D rendering using other Khronos APIs."

[http://www.khronos.org/egl/]

OpenGL on EGL

?

Platform

x11 / directFB / linuxFB / wayland

e.g.: EGL on X11

- Display display = XOpenDisplay(NULL);
- connection = XGetXCBConnection(display);
- xcb_create_window(connection,…, window,…)

... back to EGL

- "mapping" display
  - EGLDisplay eglDisplay = eglGetDisplay(display)
- creating EGLSurface
  - eglCreateWindowSurface(eglDisplay, config, window,...);
- creating EGLContext
  - eglCreateContext(display, config,...);
- preparing painting
  - eglMakeCurrent(display, surface, surface, context);

Different EGL implementations!

EGLConfig

1. create native window and extract native window-format
   (bpp, alpha, ...)

2. eglGetConfigs => a list of EGLConfigs

3. find the EGLConfig which matches your native format
   ...and application specific needs

Qt makes it simpler

Platform

x11 / directFB / linuxFB / wayland


windowing system

choosing a window system

- the quality of the platform integration is vital!
- An incomplete list of windowing systems:
  - no windowing system / LinuxFB
  - X11
  - DirectFB
  - Wayland

platform-plugin

	OpenGL	multi-surface OpenGL
xcb	✓	✓
wayland	✓	✓
directfb	?	?
eglfs	✓	x
minimalegl	✓	x
kms	✓	x
minimal/linuxfb	x	x

runtime: -plugin <plugin-name>, QT_QPA_PLATFORM

compile time: QT_QPA_DEFAULT_PLATFORM <plugin-name>

X11

- pro
  - matured and very well tested code
  - extremely stable API
  - wide adoption
  - extensions
- con
  - matured, even archaic architecture
  - originally meant for different use-case and technology
  - complex code base
    ... which very few people are familiar with
  - hard to optimized

DirectFB

- supposedly pro
  - fast IPC and compositing
  - additional 2D-painting API
  - rather flexible
- con
  - limited adoption / limited HW support
  - only a hand full people are familiar with the internals
  - additional 2D-painting API -- it's a bit alien to Khronos APIs
  - unknown QPA-plugin state

Wayland

- pro
  - the newest and most modern architecture
  - bound to use HW-accelerated compositing
  - EGL based: open to any Khronos API!
- con
  - quite new
  - still moving API
  - not complete in all regards

… it's the future! :)

no windowing system

- pro
  - by far the leanest graphics-stack
    - smallest footprint
    - fastest pipeline
  - simple to setup and to get running
  - typically very good HW support
- con
  - single-process, single window
  - possibly proprietary/specific API
  - stability and security

EGLFS

- a single process, single surface platform plugin
- has support for many things (e.g. mouse cursor)
- has become rather complex by now
  … minimalegl is the new eglfs
- behavior can be mkspec-specific via QEglFSHooks

QEglFSHooks

- API to alter, specify or fill missing behavior of the eglfs(!)-plugin
  - qtbase/src/plugins/platforms/eglfs/qeglfshooks.h
- sub class QEglFSHooks and provide alternative implementation
- part of device-mkspec
- qmake.conf: EGLFS_PLATFORM_HOOKS_SOURCES=$$PWD/file.cpp

Appendix

GLX

"OpenGL Extension to the X Window System"

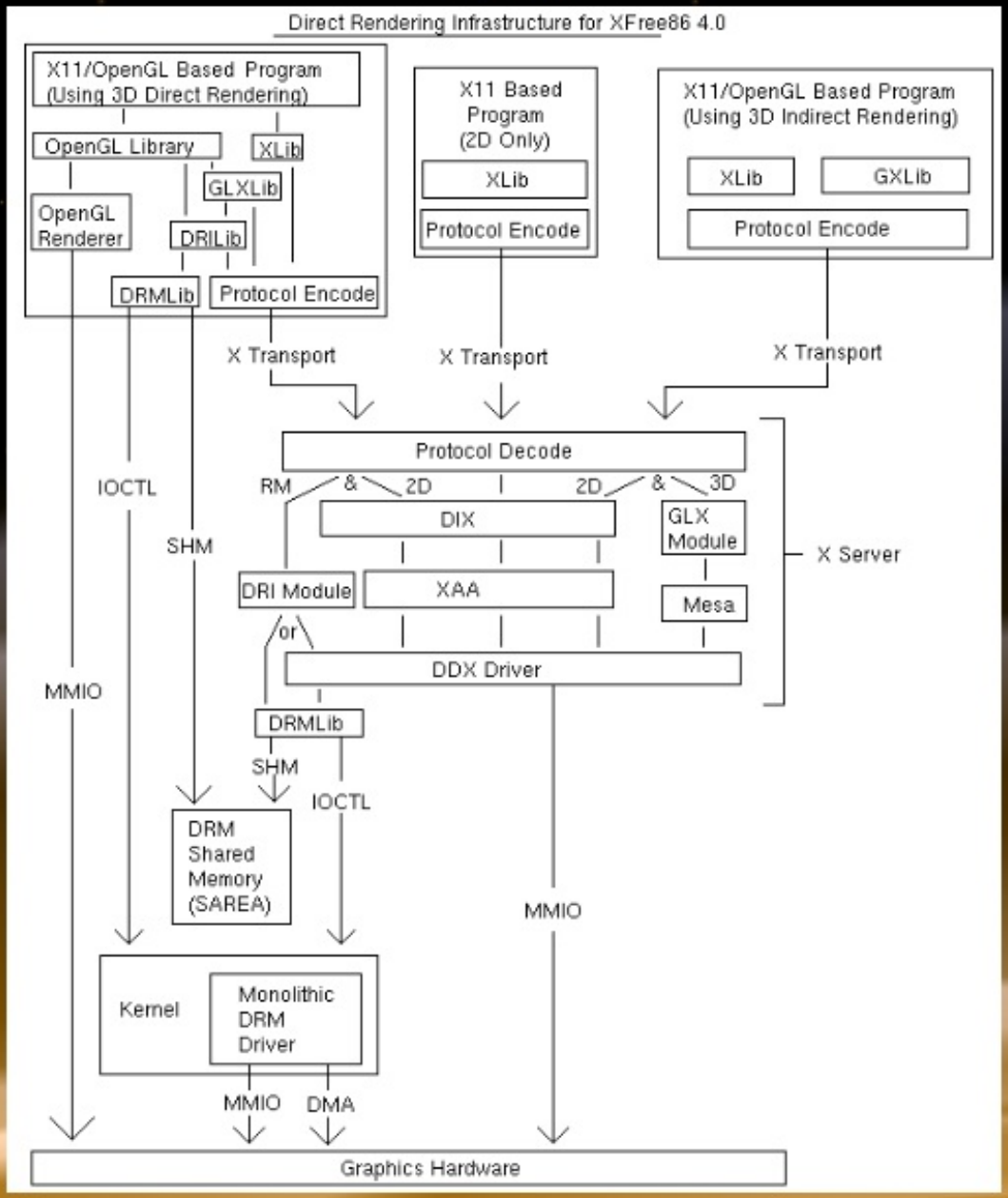
OpenGL platform integration => alternative for EGL

DRI

In computing, the Direct Rendering Infrastructure (DRI) is an interface and a free software implementation used in the X Window System to securely allow user applications to access the video hardware without requiring data to be passed through the X server.

[en.wikipedia.org/wiki/Direct_Rendering_Infrastructure]

DRI



[http://www.linuxgraphics.cn/xwindow/dri_hack.html]

DRM

Direct Rendering Manager

kernel module (and userspace lib) to manage GPU resources

KMS

Kernel Mode Settings

Mode setting is the setting up of the screen resolution and color depth mode for a computer graphics card.


Thank you