SONY

Embedded Linux UI Comparison

Tim Bird
Senior Staff Software Engineer
Sony Electronics

Agenda

- Embedded Linux UI options
- Comparison points
- Presence at ELC
- Evaluations
- Resources

Embedded Linux UI options

Embedded Linux UI options

- Qt
- Enlightenment Foundation Libraries
- GTK
- Kinoma
- Crosswalk
- FLTK
- DirectFB
- Nano-X

Narrowing the field

- Qt
- Enlightenment Foundation Libraries
- GTK
- Kinoma
- Crosswalk
- FLTK
- DirectFB
- Nano-X

Low-level, older UI systems

- FLTK – Fast Light Toolkit
  - Still in development, but a very small community
  - Popular in 2003/2004, but hasn't grown a lot since then
- DirectFB
  - Was popular for a while, got CE Linux Forum funding
  - Work appears to have stopped about 2 years ago
  - Community never hit critical mass
- Nano-X (Microwindows)
  - Very low level – really an X replacement for low end
  - No widgets, themes, IDE, designer, etc.
  - Last active development in 2010

The contenders

- Qt
- Enlightenment Foundation Libraries (EFL)
- GTK
- Kinoma
- Crosswalk

Comparison Points

Comparison Points

- Language
- Features/functionality
- Performance
- License
- Community size
- GPU driver support
- Platform support
- Ease of development
- Resource requirements

Language

- What is the native language?
  - C, C++, Javascript
- What additional bindings are supported?
  - Python, Perl, Javascript, other scripting languages?

Features/functionality

- All will have drawing primitives, events, widgets
  - What higher-level elements are available?
- What other features:
  - Data type primitives
  - Theming
  - Scripting
  - Video integration
  - Web/HTML support
  - Internationalization
  - Threading
  - IPC

Performance

- How fast does it run?
  - Frames per second
  - Number of objects
  - UI responsiveness
- Does it utilize the GPU?
- How efficient is it?
  - Can it be used on low-end processors?
- Does it utilize multi-core CPUs

License

- Broad categories:
  - Proprietary
  - Permissive
  - Copyleft
- BSD, Apache, MIT, LGPL 2, LGPL 3, GPL 2, GPL 3
- License is important!
  - Some projects may not be possible with LGPL/GPL 3
- Stability of license
  - Can it change?
- Are there contributor agreements?

Community size

- How big is contributor base?
- How active is the community?
- How big is the user base?
- Likelihood of long-term maintenance and improvement
  - Don’t want to get stranded on a dead project

GPU driver support

- What underlying graphic systems are supported
  - Frame buffer
  - DRM
  - X11, Wayland
  - OpenGL ES
- Who does the GPU driver work?
  - CPU/GPU vendor?
  - Sony?

Platform support

- What operating systems does it run on?
  - Linux, Windows, Mac
  - BSD? (could SCE use it?)
  - Android, IOS
- What operating systems can you develop on for it?
- Is it also used for non-embedded (ie desktop Linux)?
- These factors affect the size of the community

Ease of development

- UI design tools
- IDEs
- Documentation
- Samples

Resource requirements

- Size
  - Disk space for system
  - Disk space per app
  - Memory for running app
- CPU/GPU
  - What processor and speed are required to perform adequately?

Presence at ELC

Presence at ELC

- EFL
- Kinoma

EFL notes (highlights)

- EFL does a scene graph
  - Is not a dumb canvas
  - EFL is full-featured, but can scale down, if needed
- Tizen supports both HTML5 and native apps (EFL)
  - However, mobile SDK has a weird license (Flora)
- Enlightenment was designed for embedded
  - More optimized for battery and memory than others

EFL notes (2)

- Project switched to binary releases
  - When they switched, about 90% of people stopped building from git.
- Community
  - Is centered in Europe
  - 80% of commits are from Samsung
  - 2 companies do commercial support for EFL and enlightenment
    - In Brazil and France

EFL notes (3)

- Refining their rendering pipeline
  - Take into account modern hardware with multiple CPUs
- Claim that EFL is highly optimized
  - Can optimize rendering due to scenegraph
  - For example, EFL does partial updates for OpenGL
- Have theming – optimized for load time
  - Time to first frame is short
- Widgets take into account scale factor and finger size
  - For multiple device form factors (DPI is not good enough)
    - From watches to TVs

EFL notes (4)

- New Eo project
  - Fresh API that is cleaner
  - Automatic binding generation
  - Some documentation automation
  - EFL 2.0

Kinoma notes

- Kinoma JS = Javascript-based UI system
- Has nice-looking graphics
- Has a simple system to deploy to target from host
- Have their own Javascript engine (XS6)
- They claim performance is good
  - Interpreter doesn’t do blitting or rendering, just logic and layout
- Not much community
  - Kinoma JS is basically supported by one company (Marvel)
    - 2 developers, 95 commits, 30 github forks

Evaluations

Qt Basics

- Language = C++
  - Bindings for Python, Javascript, and many more
- Features/functionality
  - Support for native code and scripted
  - Has rich support for just about everything
    - Themes, Data types, Video, Web/HTML, Internationalization, Threading, etc.
  - Has look of native OS (for desktop apps)
- Performance
  - Good – haven’t done my own measurements

Qt License

- Older versions = LGPL 2.1/GPL 3, and proprietary
  - Until Qt v.5.3
  - Qt v.5.4/5.5/5.6 : Newly added features → LGPL v.3
- Latest version= LGPL 3/ GPL 2, and proprietary
  - From Qt v.5.7 (May 2016)
- License is not stable
  - Qt Company can change it (for new releases)
- Requires code to be submitted under a contribution agreement
- New license creates a problem for companies that wish to avoid GPL 3

Qt Proprietary license pricing

- Proprietary license is very expensive
- Minimum of $250,000 + $11 per unit
  - But they won't tell you that up front
- No pricing transparency
  - Each price is determined in negotiation with Qt Company on a per-project basis
  - Developer seats can't be transferred from one project to another
- There is no corporate buyout option
  - That is, there's no option for a one-time buyout

Qt Community size

- Very large
  - Developers – hundreds
  - Users – thousands
- Mail list traffic
  - 200-500 messages monthly on qt-dev
- Used by many companies
- Used as basis for KDE (desktop Linux)

Qt GPU driver support

- X11
- Framebuffer
- Wayland (in progress)
- OpenGL ES

Qt Platform support

- Linux, Windows, Mac
- BSD
- Android
- IOS

Qt Ease of development

- Very mature tools
  - UI design – Qt Quick Designer
  - IDE – Qt Creator
- Lots of documentation
- Coding style and API design guidelines
- Lots of examples

Qt Resource requirements

- Unknown
  - Suspect it requires more resources than the others
- I need to measure this

EFL Basics

- Language = C
  - Bindings: Python, Javascript (in progress)
- Features/functionality
  - Themes = Edje
  - Data Types = Eina
  - Scripting = Embryo
  - Video = Emotion
  - IPC = EET
  - And more…

EFL Performance

- Claims to be good – haven’t been able to measure yet

EFL License

- BSD + LGPL v2
- Cannot change
- No contributor agreement

- (These are all good)

EFL Community size

- Medium
  - Developers – tens
  - Users – hundreds
- Mail list traffic:
  - Have about 250 messages per month now on Enlightenment-devel
    - Mail traffic peaked in 2012 with over 1000 messages per month
- Used by some companies
  - Used by Samsung for many, many products
    - From watches to TVs
- Is native UI API for Tizen

EFL Community notes

- 80% of commit traffic is Samsung
- Is part of Tizen
- Samsung has used it in millions of devices
- Samsung looking for other companies to join in development and use

EFL GPU driver support

- Framebuffer
- X11
- Wayland (in progress)
- OpenGL ES

EFL Platform support

- Linux, Windows, Mac
- Android (in progress)
- Port to RTOS (in progress)

EFL Ease of development

- Hard to assess
  - UI designer is new (not 1.0 yet)
  - IDE is new (not 1.0 yet)
  - Progress developing these has been slow
- There are some examples
- Documentation is a bit spotty
- Some reports of type safety issues and poor error messages that make development hard

EFL Resource requirements

- Claims to be light-weight
  - Was developed with embedded in mind
- Need to measure

Kinoma Basics

- Language = Javascript (no other bindings)
- Features/functionality
  - Has own OS abstraction layer (types, methods, etc.)
- Performance
  - Unknown
  - Claimed to be good, but low-level graphics layer only has 3 primitives:
    - Fill rectangle, blit bitmap, draw text

Kinoma License

- Apache 2.0

Kinoma Community size

- Quite small
  - 2 developers
- Mainly driven by a single company (Marvell)

Kinoma (cont.)

- GPU driver support
  - Whatever is supported by browser
  - OpenGL work is in progress
  - Not sure what acceleration is supported
- Platform support
  - Whatever is supported by browser
  - Linux, Windows, Mac, Android, IOS (assume BSD, but don’t know)

Kinoma Ease of development

- Appears to be good
- Has IDE (Xcode?), with deploy to target
- Can use graphical language building block language
  - Like Scratch

Kinoma Resource requirements

- Not sure about UI requirements
  - Assume to be high because of browser
- For IOT (headless) use XS6 JavaScript Engine on device
  - 200 MHz ARM Cortex M4
  - Saves RAM by using XIP to run native ARM code directly from flash memory, and byte code from flash memory

GTK Basics

- Language = C
  - Bindings: C++, Python, Javascript and many more
- Features/functionality = Rich?
- Performance = Good?
- License = LGPL 2
  - Unlikely to change

GTK more

- Community size
  - Desktop community is large
    - Hundreds of developers
    - Thousands of users
  - Embedded community is unknown size
- GPU driver support
  - X11, framebuffer, OpenGL (recently)
- Platform support – Linux, Windows, Mac
- Ease of development - ???
- Resource requirements – Need to measure

Crosswalk Basics

- Language = HTML5 + CSS + Javascript
  - no other bindings
- Features/functionality
- Performance - ???

Crosswalk More

- License – LGPL 2
- Community size
  - Unknown, used in Tizen for non-native apps
  - Supported primarily by Intel
- GPU driver support – whatever the browser supports
- Platform support – Linux, Windows, Android (more?)
- Ease of development - ???
- Resource requirements - ???
  - Assume to be high, because of browser

Planned investigations

- Port test app to all platforms, to measure resource requirements and performance
  - Bouncing ball

Resources

Resources

- Thomas Petazzoni presentation (2008)
  - http://elinux.org/images/6/64/Choosing-embedded-graphical-libraries.pdf
- https://en.wikipedia.org/wiki/List_of_widget_toolkits#Comparison_of_widget_toolkits
- http://wiki.lxde.org/en/GUI_Toolkit_Comparison (from 2013)
- https://en.wikipedia.org/wiki/Comparison_of_X_Window_System_desktop_environments
- http://bird.org/sony/UI_Comparison_Table
- Cedric Bail presentation on EFL (2016)
  - Slides: http://elinux.org/images/8/84/ELC2016-bail.pdf
  - Video: https://www.youtube.com/watch?v=S062ft-BYsg

Done.