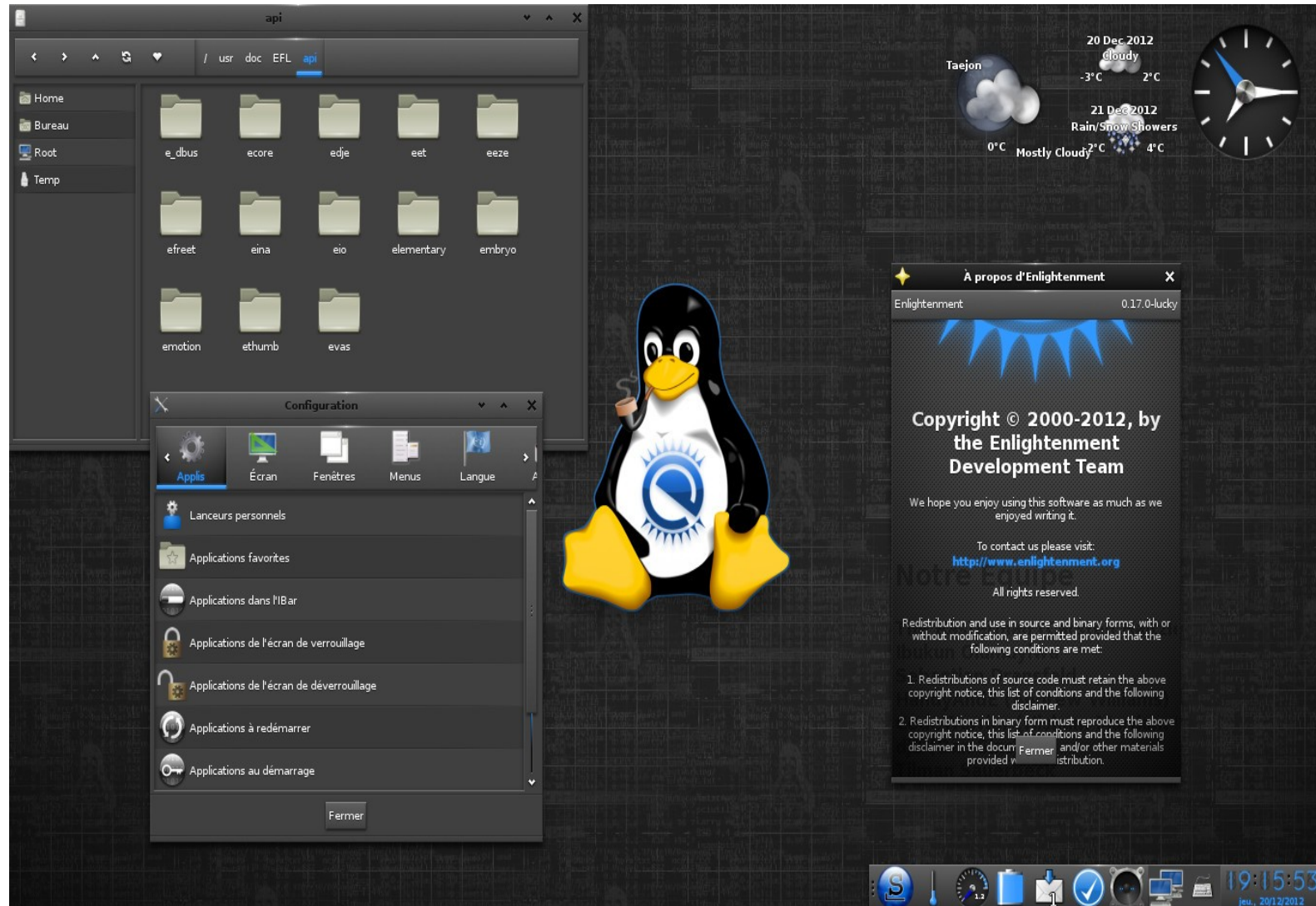
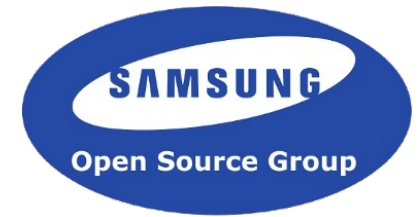


# EFL at ELC 2016

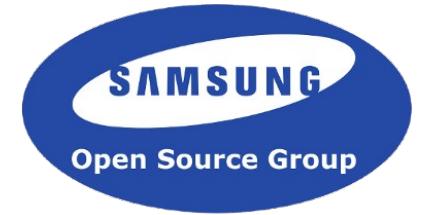
A toolkit for developing efficient and attractive  
Uis on embedded devices

Cedric BAIL  
Samsung Open Source Group  
[cedric@osg.samsung.com](mailto:cedric@osg.samsung.com)

# EFL: A Toolkit Created for Enlightenment 17

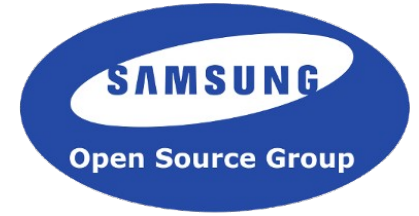


# Enlightenment 17



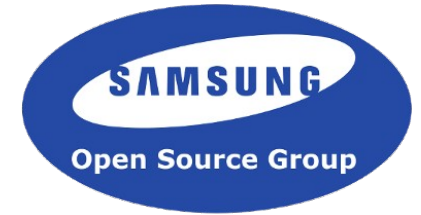
- Enlightenment project started in 1997
- Windows Manager
- First Windows Manager of GNOME
- Full rewrite started in 2001
- Primary belief is there will never be *“a year of the Linux desktop”*
- Designed with the embedded world in mind...
- ... and needed a toolkit !
- As none matched our need back then and still don't today !

# Enlightenment



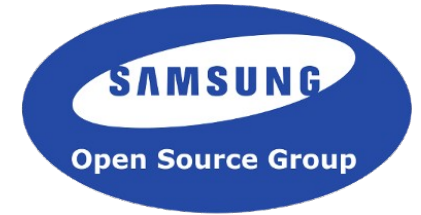
- 17 was just the version number, we are now at 20!
- Use EFL scenegraph and main loop
- Not different from any other EFL application
- Composite and Window Manager
- Wayland client support
- Multiple backend (X, FB, DRM, ...)
- Highly customizable (Profiles, modules and themes)

# Enlightenment Community



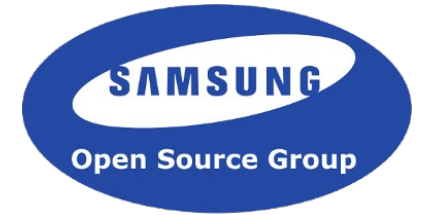
- The Enlightenment community
  - 60 uniq contributors per release (10 cores)
  - 1000 users that build from source
  - 2 distributions based on Enlightenment (Bodhi and Elive)
- The Enlightenment community expected Linux to takeoff in the embedded world, not on the desktop
- Multiple company using it around the world (Mostly Samsung and small company in France)
- Companies providing support for it (Brazil and France)
- The values shared by this community:
  - Fast
  - Customizable
  - Light
  - Scalable
  - Feature Rich

# Enlightenment Foundation Libraries (*EFL*)



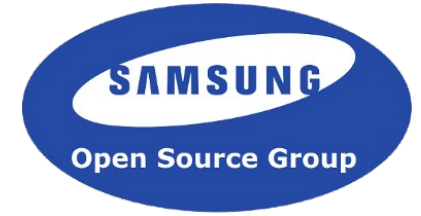
- Spent a decade writing a modern graphic toolkit
- Licensed under a mix of LGPL 2.1 and BSD license (Yes, nobody own it, nobody can change the license)
- Focus on embedded devices
- Used in Samsung Tizen product
- First release on January 2011
- Stable, long term API/ABI
- In the process of releasing version 1.13
- 3 month release cycle

# State of EFL



- Designed for creating a Windows Manager (WM), now used for any type of application
- Has its own scene graph and rendering library
- Optimized to reduce CPU, GPU, memory and battery usage
- Supports international language requirements (LTR/RTL, UTF8)
- Supports all variations of screens and input devices (scale factor)
- Supports accessibility (ATSPI)
- Fully Themable (layout of the application included)
- Supports profiles
- Can take up as little as 8MB of space with a minimal set of dependencies
- Has a modular design

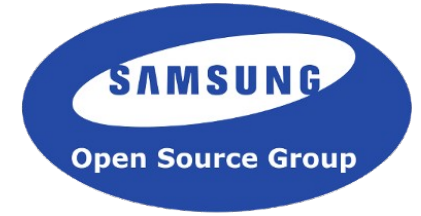
# Why We Care About Optimization



- Moore's law doesn't apply to battery and memory bandwidth
- Anyway Moore's law is slowing down !
- Most rendering operations are limited directly by memory bandwidth
- Many embedded devices have less available memory than a low end phone
  - Refrigerator, oven, dish washer, washing machine, home automation...
- Even a low end phone doesn't have much memory to spare once you run a browser!
- GL context at best consumes 10MB, usually more around 40MB; this is bad
- for multitasking!



# Current State of Optimization

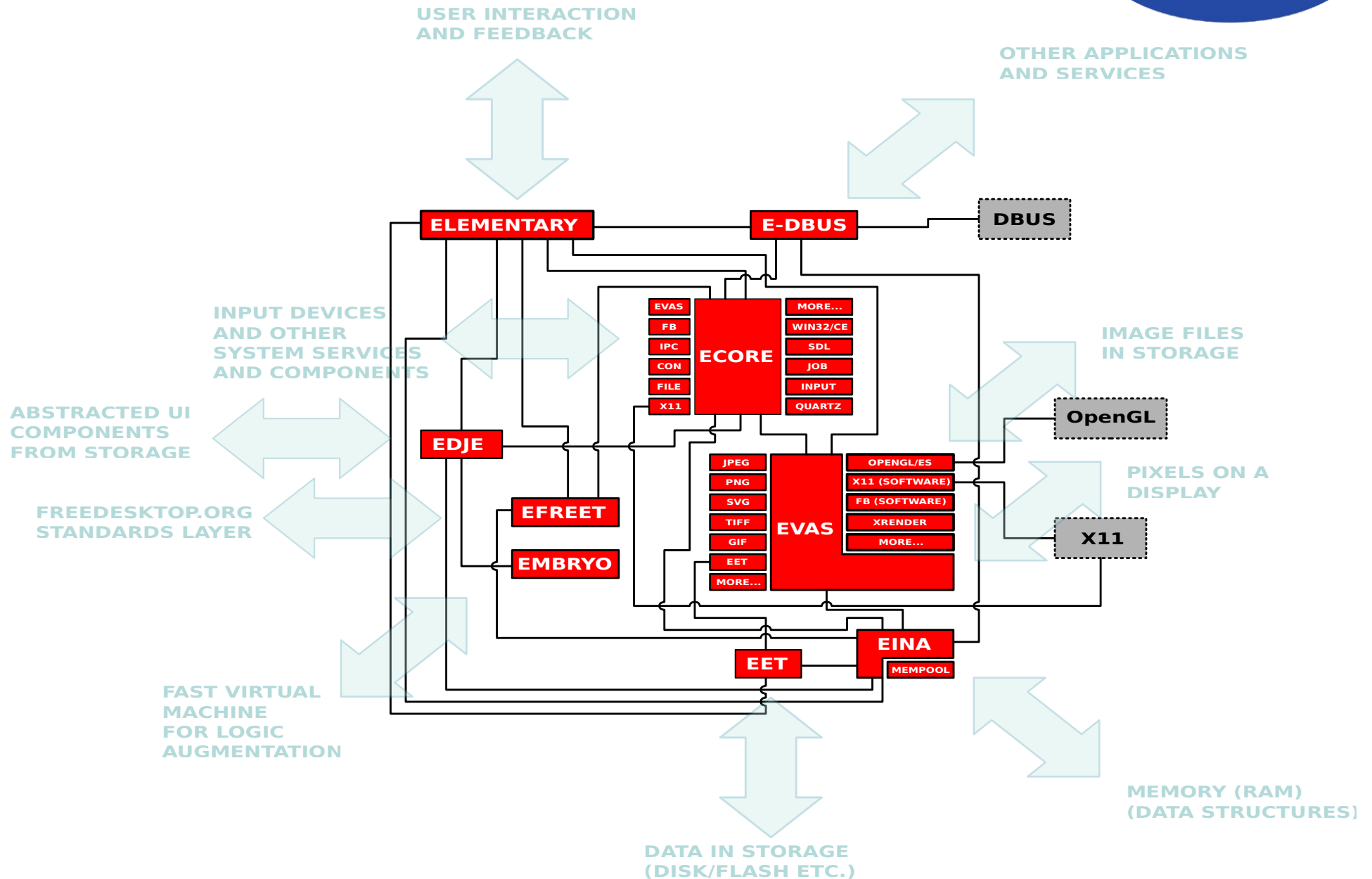
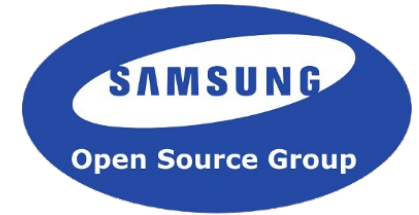


- Application runtime memory use is mostly driven by screen size
- EFL can fit in 8MB on disk (static compilation with minimal dependencies)  
(With Elementary our widgets set)
- Minimal size for just a scenegraph and a minimal set of dependency <1MB  
(Up to Ecore\_Evas with jpeg/png, but no edge support)
- No hard requirement on the GPU
- Enlightenment + Arch Linux combined :
  - 48 MB RAM
  - 300 Mhz (1024 x 768)
  - Yes, for a desktop profile!

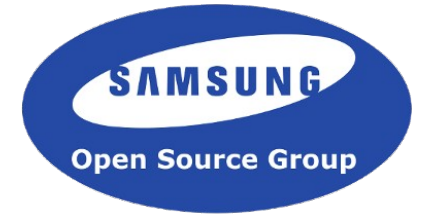
# What does EFL do for you ?



# EFL – Big picture !



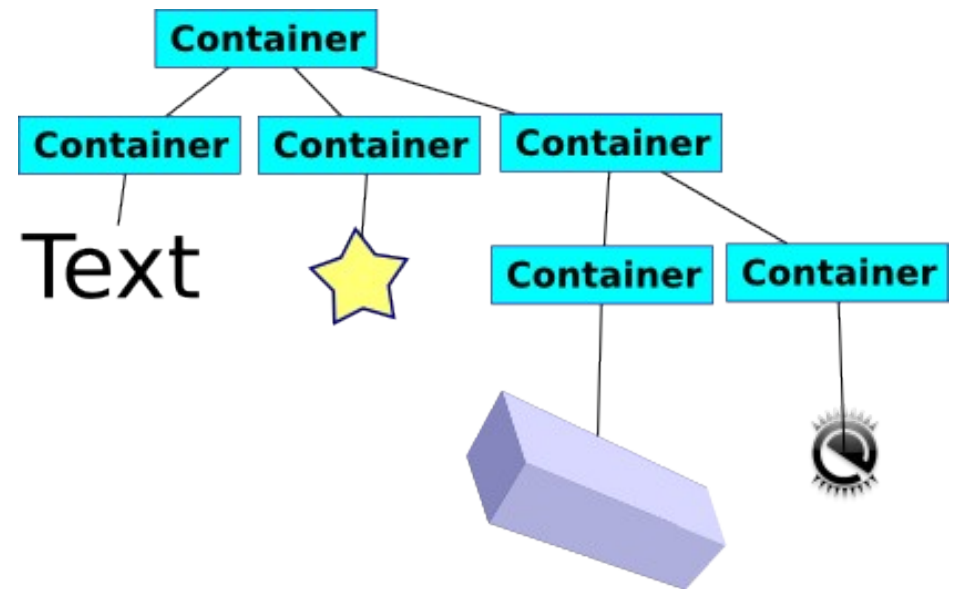
# Evas



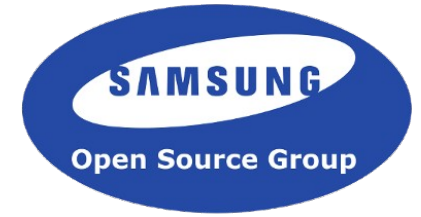
- The brain of EFL
- Scene graph library with more than 15 years of optimization in it
- Glyph free rendering
- Reduce overdraw
- Reduce waste of memory by deduplicating as much as possible
- Compressed glyph rendering
- Portable (SDL, X11, Wayland, FB, DRM, Windows, Mac OS X, ...)
- Optimized software renderer (MMX, SSE\*, Neon)
- Optimized use of GPU (optional)
  - Support partial update if driver do
  - Reduce context and texture switch as much as possible
  - Reduce memory overhead

# Evas - Scenegraph ?

- Graph of primitive graphical object
- Widget composed of primitive
- Doesn't rely on OS to draw widgets
- Compositor can also use them
- One place to optimize
  - Reuse data
  - Cutout
  - Limit shader and texture switch
  - Split CPU, memory and IO bound

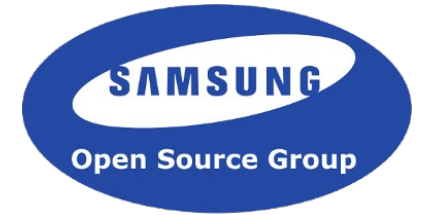







# Evas - Since early 2000



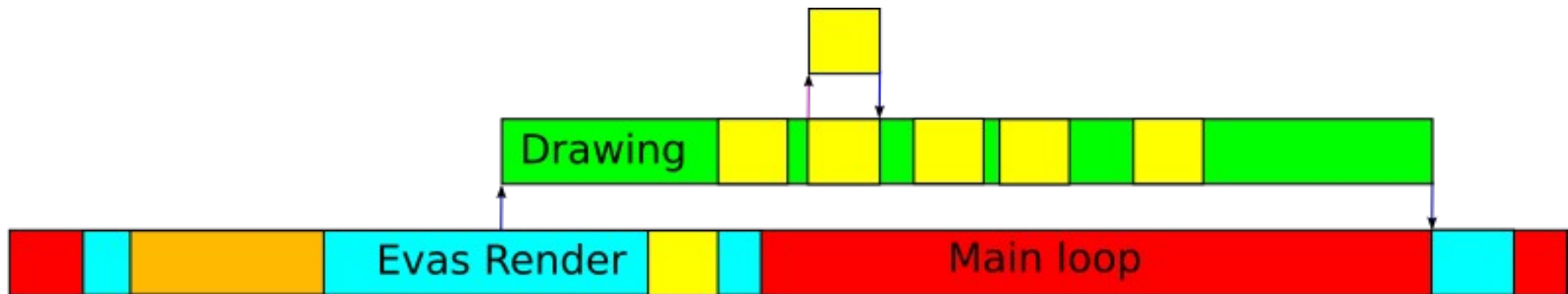
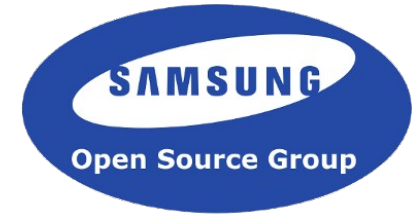
- SMP was luxury
- GPU where limited or non existent
- iOS and Android didn't exist
- It was the web bubble, nobody really did care about application on embedded device
- That's when started Evas design, nothing close for 2D toolkit






# Evas - First scenegraph



-  computing CPU intensive data
-  compositing data, mostly memory bound
-  layout object
-  walk tree to order operation
-  application logic

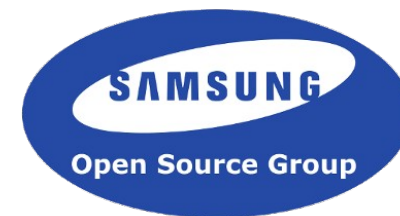
# Evas – Scenegraph today



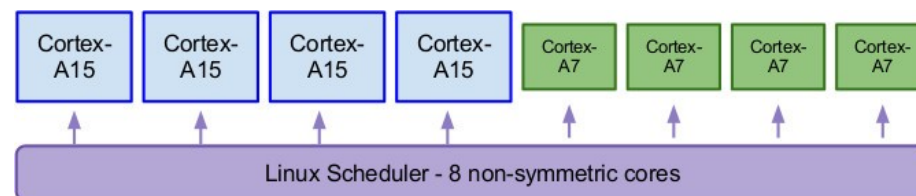
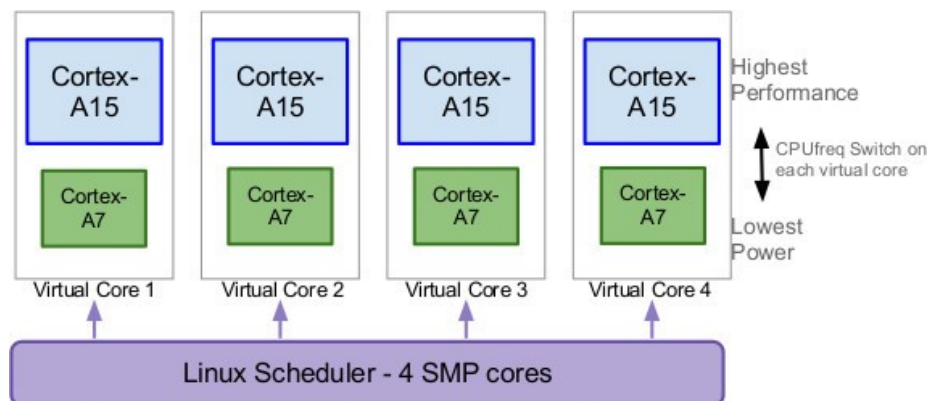
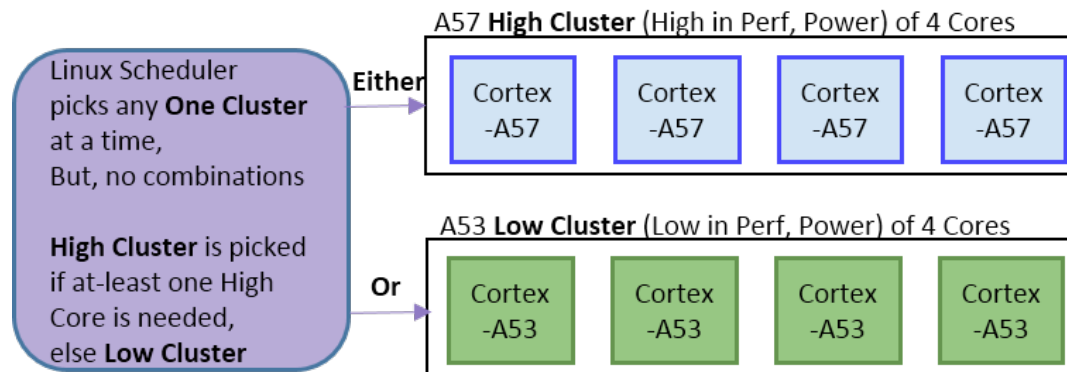
-  computing CPU intensive data
-  compositing data, mostly memory bound
-  layout object
-  walk tree to order operation
-  application logic



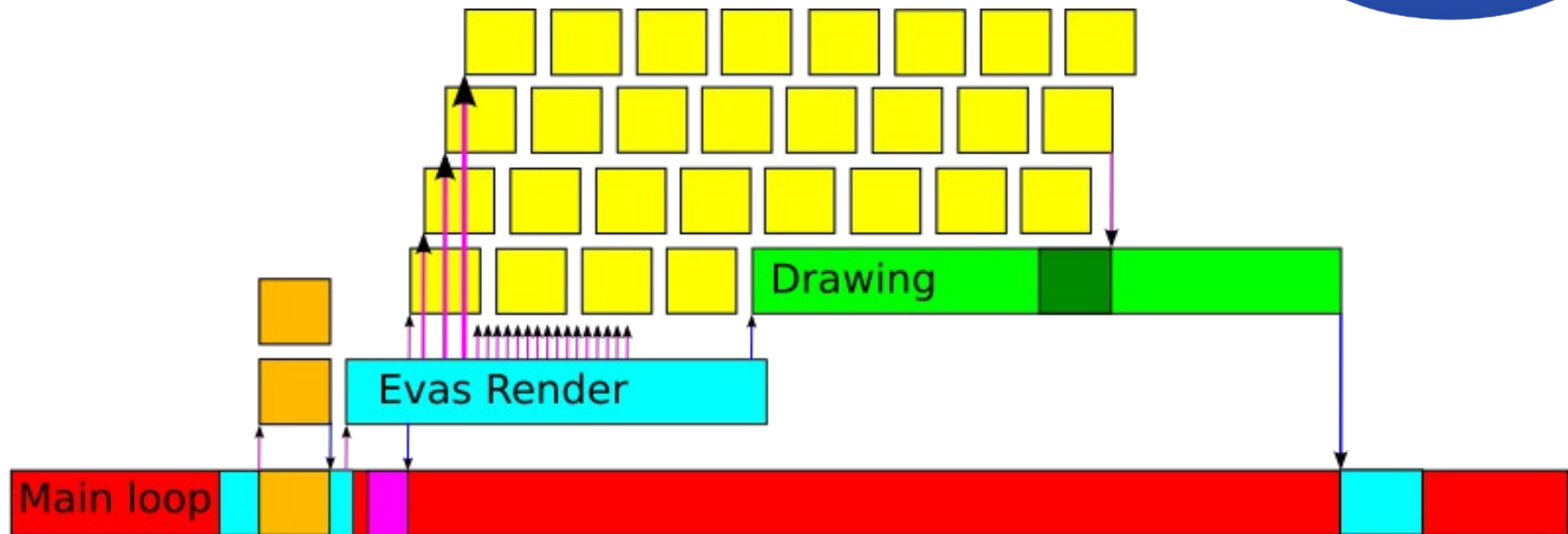
# Landscape is evolving !



- Moore's law is slowing down
- Big-Little
- Scheduling, CPU idle and frequency
- Memory bandwidth still limited
- Battery still limited
- Display everywhere

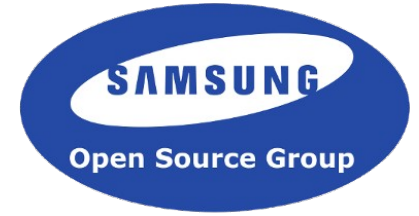


# Evas – Scenegraph tomorrow



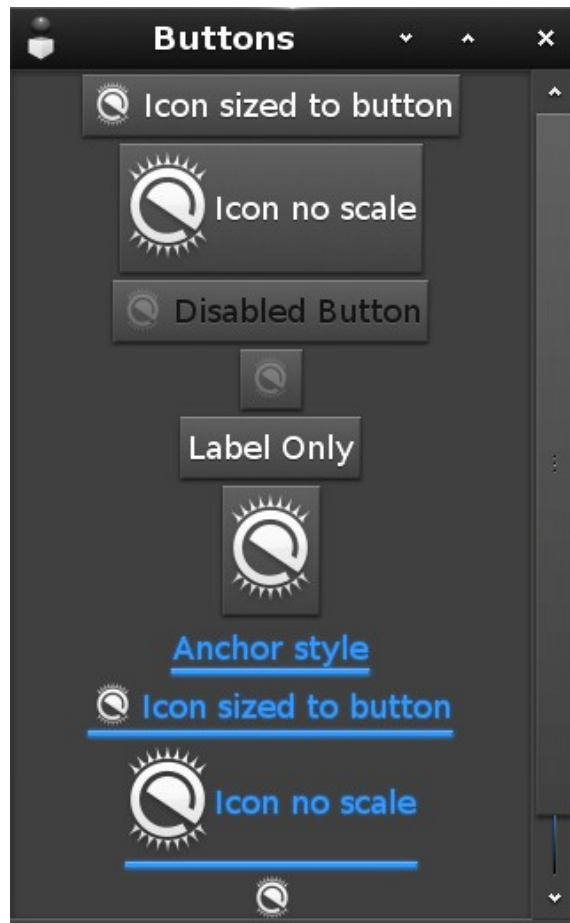
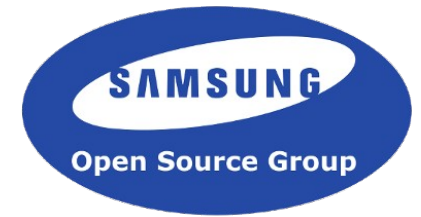
- computing CPU intensive data
- compositing data, mostly memory bound
- layout object
- walk tree to order operation
- application logic
- waiting for COW

# Edje

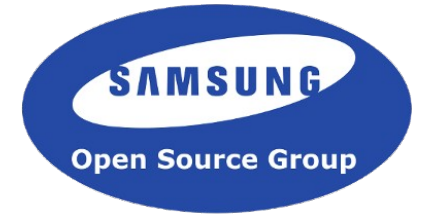


- The heart of EFL
- Theme and layout engine
- Descriptive language
- Use Evas for rendering logic (fully independent from the system)
- Doesn't require a FPU
- Optimized load time (time to first frame) and run time
- Reduced memory fragmentation

# Edje – Theme examples

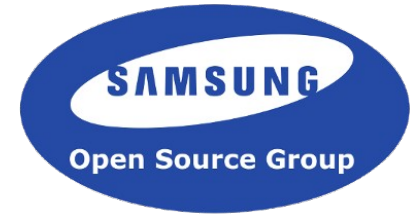


# Elementary



- Widgets toolkit
- MVC
- Use Edge and Evas infrastructure
- Screen and input independence achieved by :
  - Scale factor
  - Finger size
- Profile support (define configuration on a per Window basis)
- Fully themable
- Support touchscreen

# Release process

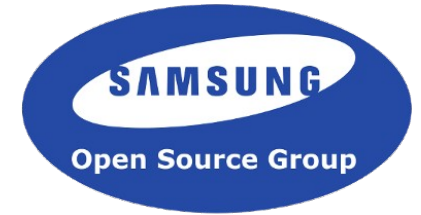


- Make check + elementary\_test pixels comparison (function coverage of all code 33.5%)
- ABI check
- API review  
(New API found by ABI check)
- Expedite performance benchmark  
(1.7 and 1.17 are as fast and consume the same amount of ressource in our tests !)
- Documentation update

# Doing your own device applications!



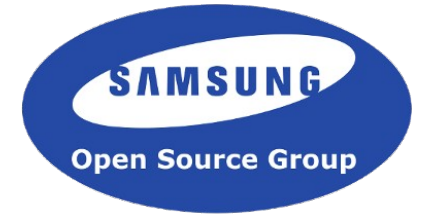
# Your own devices ?



- One or more applications ?
- Third party allowed ?
- What can your developers do ? C, JS, Python ?
- Do you have designers ?
- Long term support ?

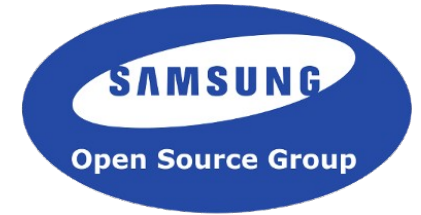


# Applications



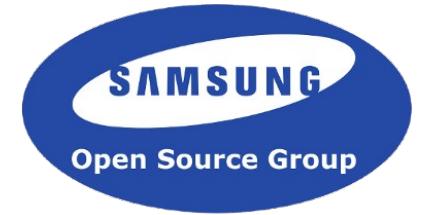
- If you intend to use multiple application, Enlightenment provide an easy to extend/customize composite and windows manager (X11, Wayland, FB)
- If you don't need to, you can directly make your application full screen using the FrameBuffer backend (or the soon to come EGL Full screen backend if your driver support it).

# Enlightenment ?!



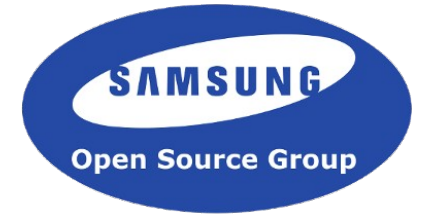
- Samsung use it in all its device as we allow application from third party everywhere (TV, Phone, Watch, Camera)
- It is a light display system that support all kind of module (windows placement, gadget, animation, ...)
- Easily configurable and support multiple profiles!

# EFL & Elementary



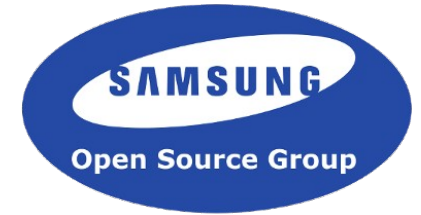
- If you intend to open development to third party, provide a theme that cover all widget that make sense on your device
- If you don't, just support what your application need.
- You can easily recycle theme between devices and extend it over time (We have tool to pick part of an existing theme)

# Configuration



- You are likely going to look at all configure options and understand what they do to limit the dependencies to your needs.
- We do not support extra configuration combination yet as nobody has taken the time to provide one that matches their needs for our VM to continually test it (If you do, we welcome such a contribution)
- That is why you will need that very long line to configure (It's just because we want you to be aware of this)

# Get involved !

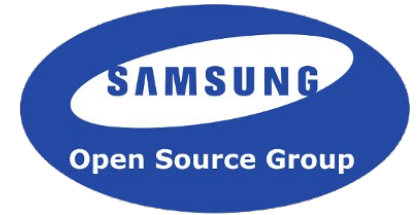


- For a one of device, you don't really need to...
- But if you plan to use EFL for longer, you should really asses your level of contribution
- Either you ask support from one of the company that work on EFL
- Or some of your engineer start to work with upstream
- The main goal is to make sure that your use case stay relevant to upstream !
- For you the additionnal benefit is to be up to date with EFL and Enlightenment progress and make sure you take all the possible benefit of it !
- Just using the toolkit won't work on the long term ! That's true for all open source project anyway...

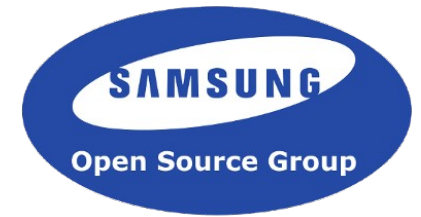
# Watches



# Cameras



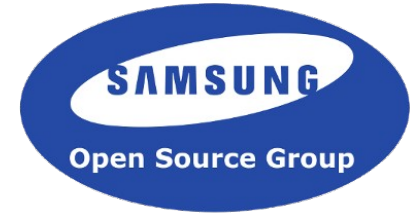
# Smartphones





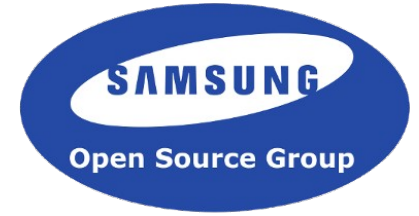


# Other devices



- Printers
- Fridge
- GPS traffic alarm
- Set top box

# Optimization for multiple application

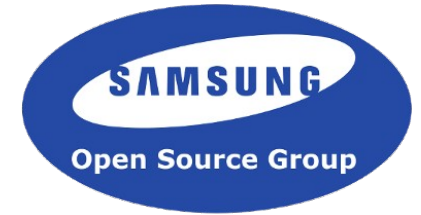


- If you have one application on screen at a time, you may maybe not be in a multiple application scenario !
- Multiple application using EFL
  - Share libraries initialization → quicklaunch  
(Faster application startup time)
- Multiple application using the same theme
  - Share fonts and image → Cserve2  
(Lower total memory footprint on the system)

# Where are we heading ?

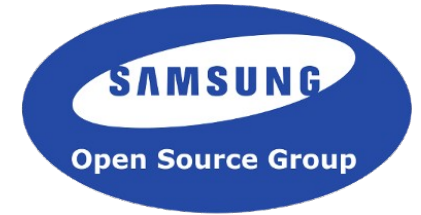


# EFL new API !



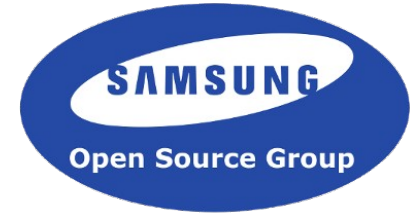
- Eo  
(a language to define portable API across language bindings at compile time)
- EFL interface  
(a new set of coherent interface to take advantage of Eo new object model)
- More emphasis on Promise and MVC
- Less work for application by providing more facility
- And you will be able to write your application in your preferred language (node.js, C++ and lua for now)

# Easier development



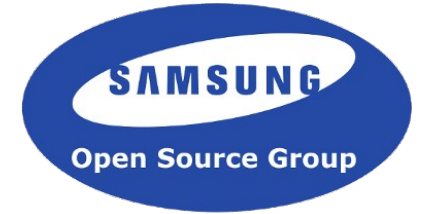
- More documentation, per language with a wiki for everyone to be able to improve it
- More tutorial/examples, thanks to Samsung contributing its Tizen documentation to upstream
- More helper in EFL, to use remote data source and database more easily in an efficient way
- More tools (Efleete, Erigo, Clouseau, Espion)
- Easier configuration of EFL (one tree, splitted Makefile for easier customization for different target)

# Improve rendering



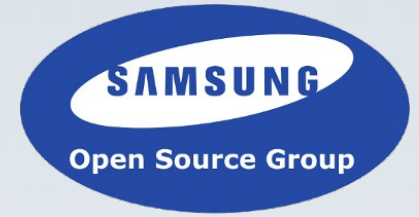
- Support vector graphics and filter using GPU
- Support sub canvas buffer and their partial update  
(Faster scrolling, able to use hardware plan)
- Asynchronous OpenGL rendering
- More parallel pipeline  
(Edje, Evas and data model)
- Next year : Vulkan !

# Community effort

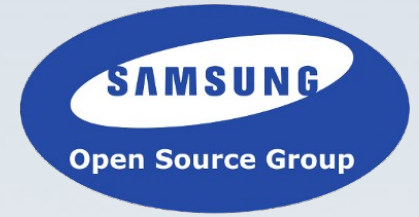


- Port to Android/libhybris and other proprietary GPU/Framebuffer OS
- Android port ?
- Python support
- IOT.js support
- RTOS port  
(main issue seems to find an open source RTOS with a supported board that does have a frame buffer !)





# Questions ?



# Thank You!