UI, Graphics & EFL

Carsten Haitzler
Principal Engineer
Samsung Electronics Korea
c.haitzler@samsung.com
Founder/Leader Enlightenment / EFL

TIZEN™
DEVELOPER SUMMIT 2014
SHANGHAI
TIZEN开发者峰会（上海）

Display System Overview

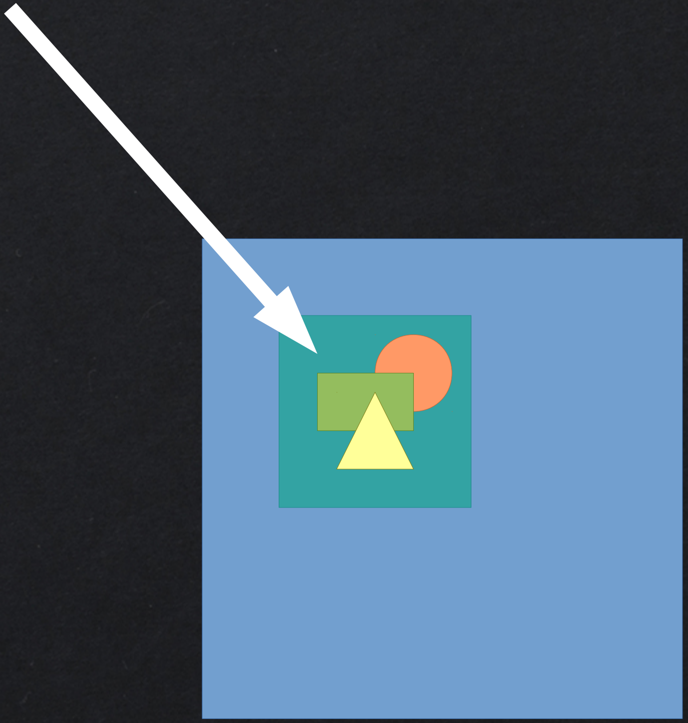
Graphics

Graphics – Old-School FB

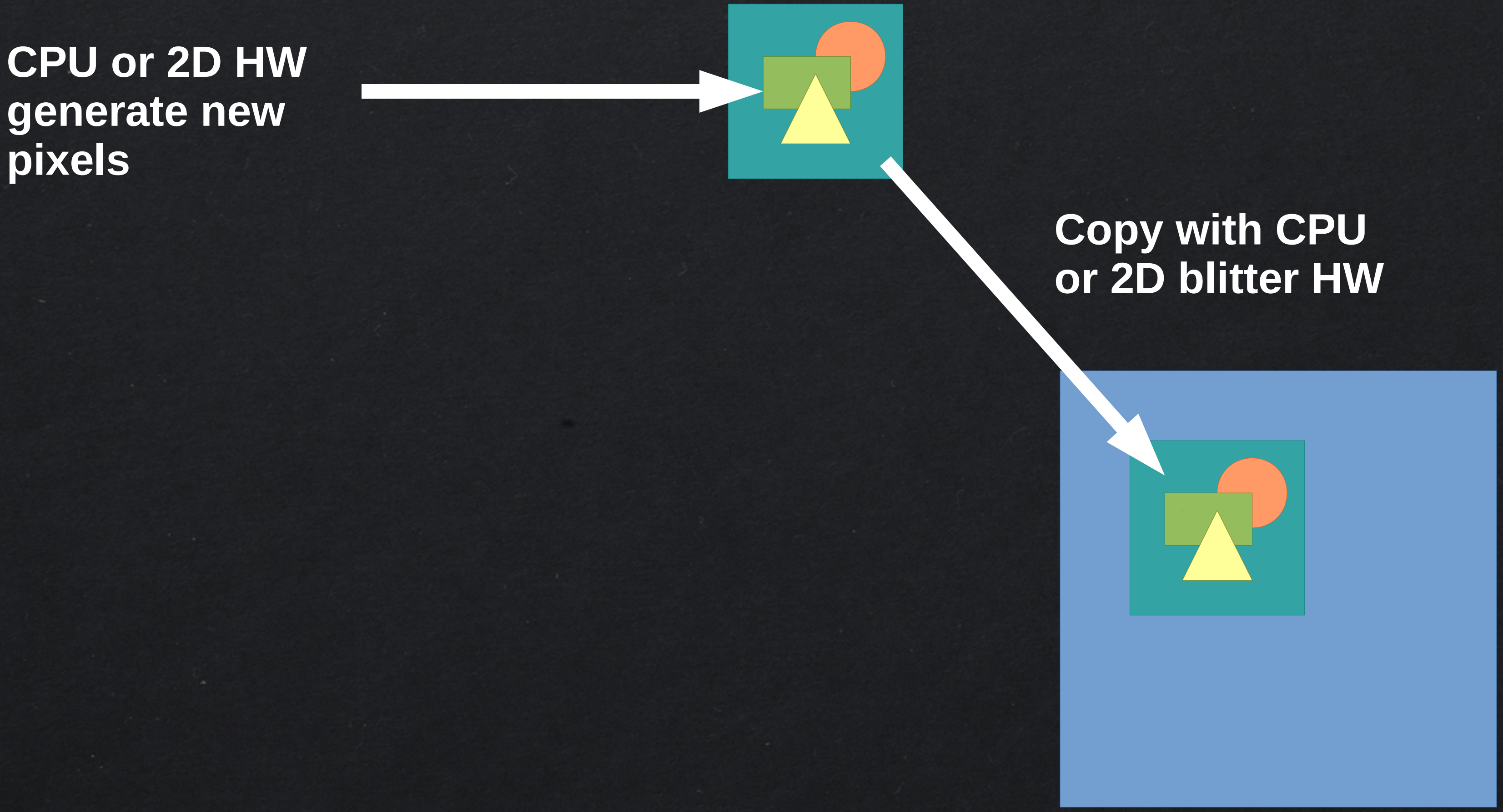
- In the old days we used the framebuffer directly
- If you do embedded work this will be familiar
- Featurephones pretty much worked this way
  - Apps “own” the screen (direct drawing)

Single Buffer - Flickering

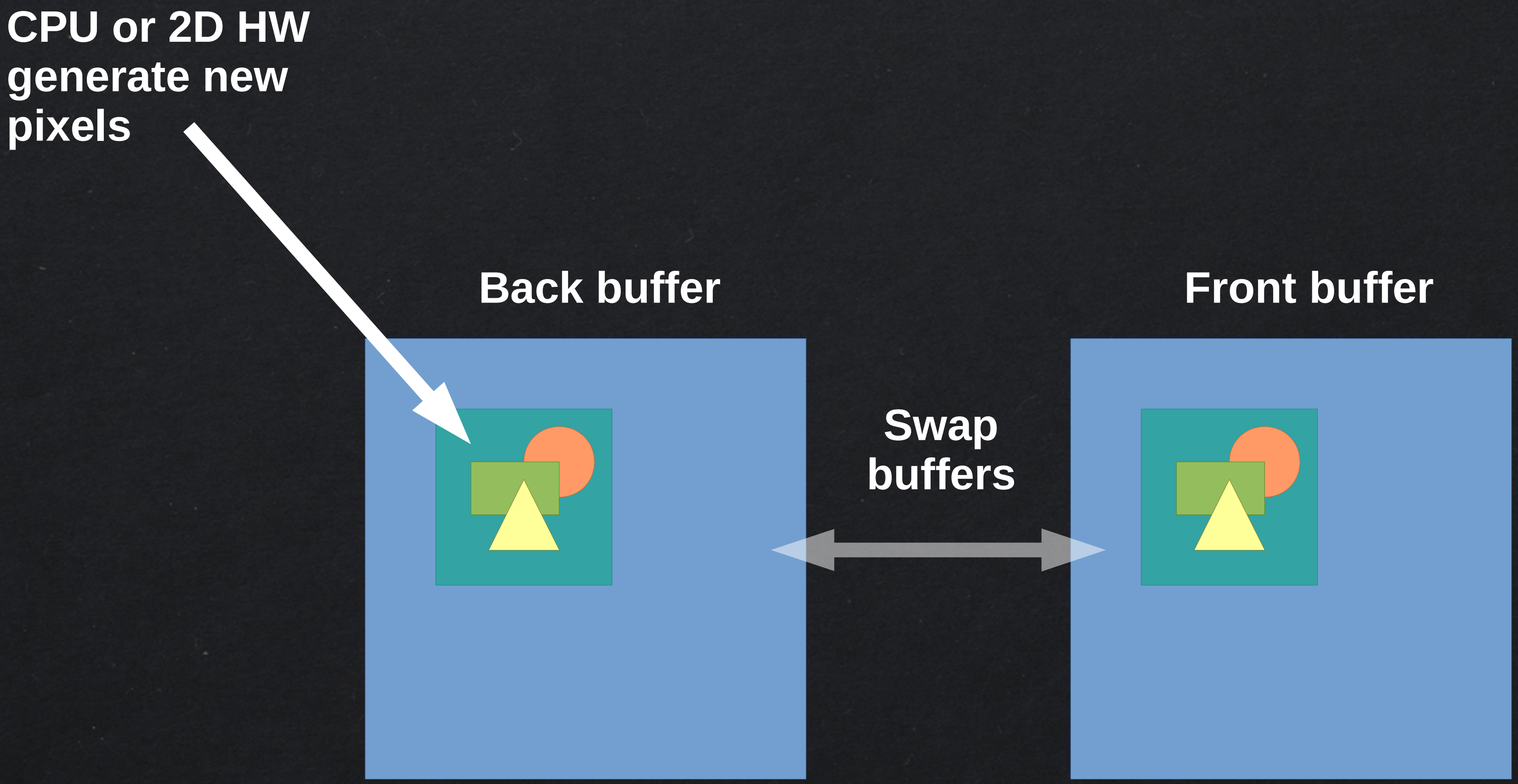
CPU or 2D HW generate new pixels / Copy around FB



Double buffer – copy region

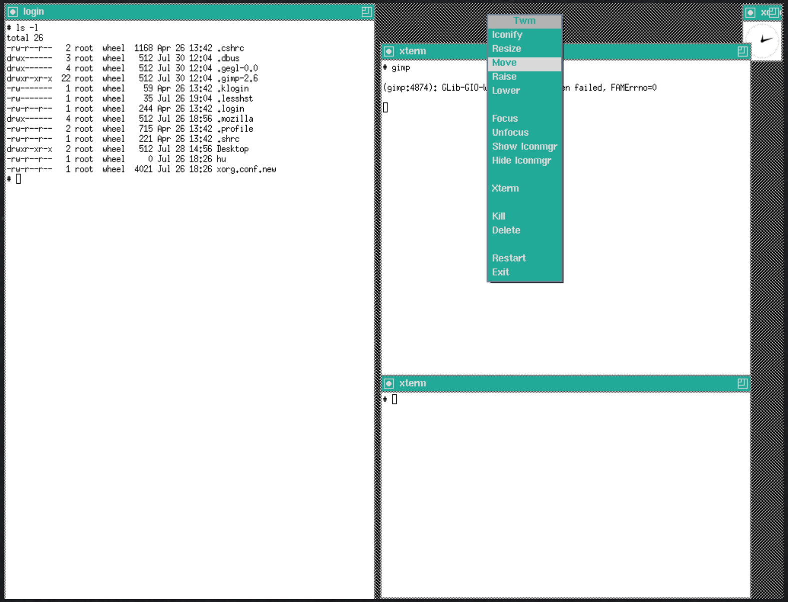

CPU or 2D HW generate new pixels
Copy with CPU or 2D blitter HW

Double buffer – swap buffers



X11 – FB Sharing (Flickering or Copies)

- Created in the 1980's to share FB
  - Also share over a network
  - Allow acceleration of operations over the network
- Multiple windows on screen at once
- Multiple screens
- Multiple bit depths at once
- Complex
- Everything is rectangles
- Drawing server-side via requests

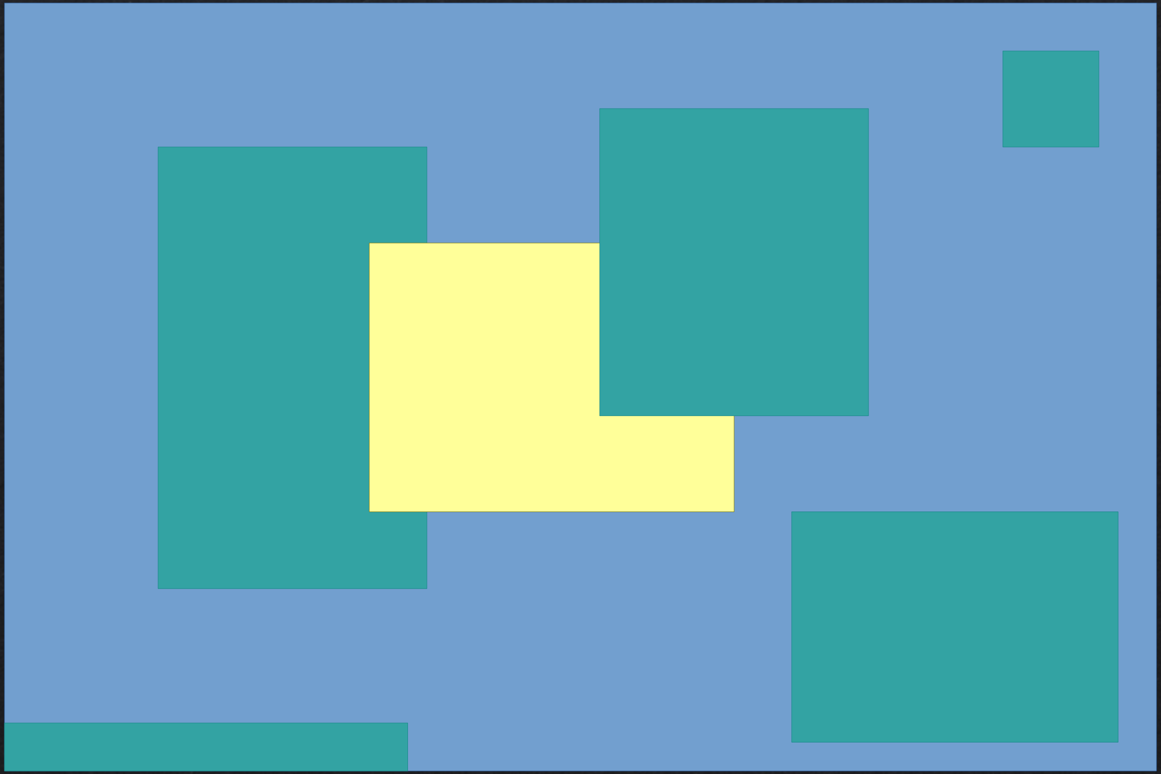


```
# ls -l
total 26
-rw-r--r--   2 root  wheel  1168 Apr 26 13:42 .cshrc
drwx------   3 root  wheel   512 Jul 30 12:04 .dbus
drwx------   4 root  wheel   512 Jul 30 12:04 .gegl-0.0
drwxr-xr-x  22 root  wheel   512 Jul 30 12:04 .gimp-2.6
-rw-------   1 root  wheel    59 Apr 26 13:42 .klogin
-rw-------   1 root  wheel    35 Jul 26 19:04 .lesshst
-rw-r--r--   1 root  wheel   244 Apr 26 13:42 .login
drwx------   4 root  wheel   512 Jul 26 18:56 .mozilla
-rw-r--r--   2 root  wheel   715 Apr 26 13:42 .profile
-rw-r--r--   1 root  wheel   221 Apr 26 13:42 .shrc
drwxr-xr-x   2 root  wheel   512 Jul 28 14:56 Desktop
-rw-r--r--   1 root  wheel     0 Jul 26 18:26 hu
-rw-r--r--   1 root  wheel  4021 Jul 26 18:26 xorg.conf.new
# 
```

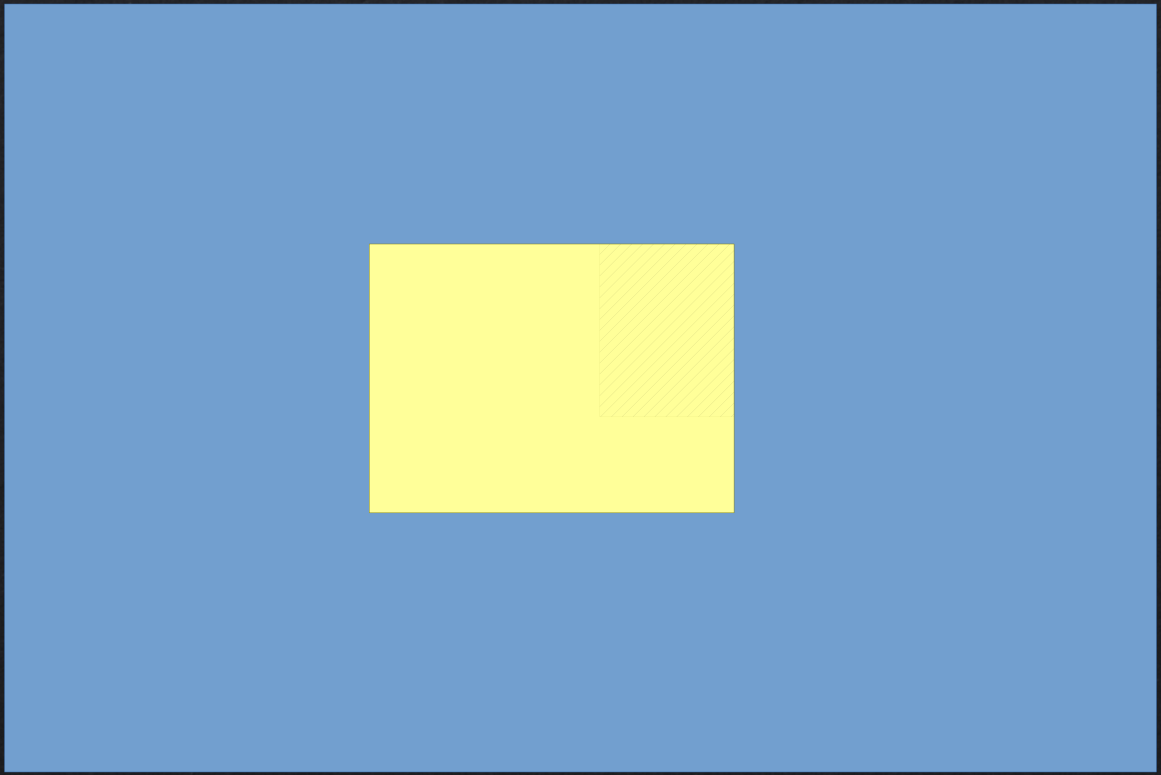
```
# gimp

(gimp:4874): GLib-GIO-W                en failed, FAMErrno=0

```

```
# 
```

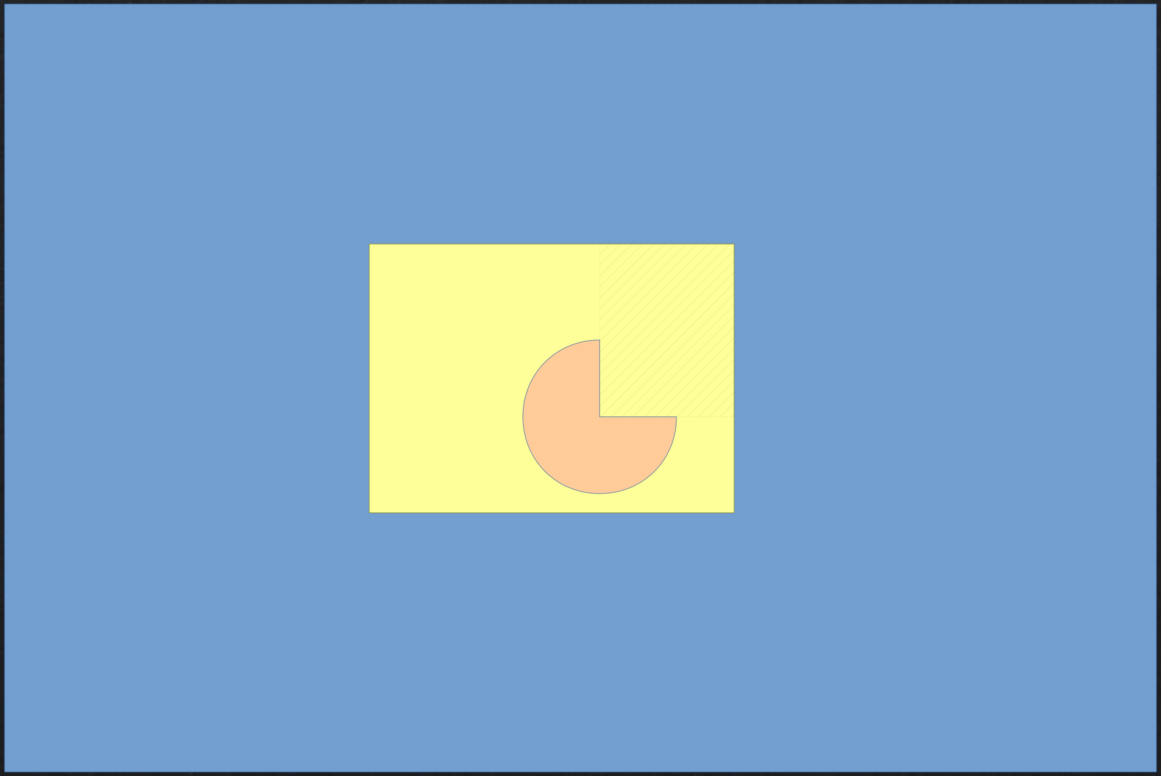
X11 Drawing



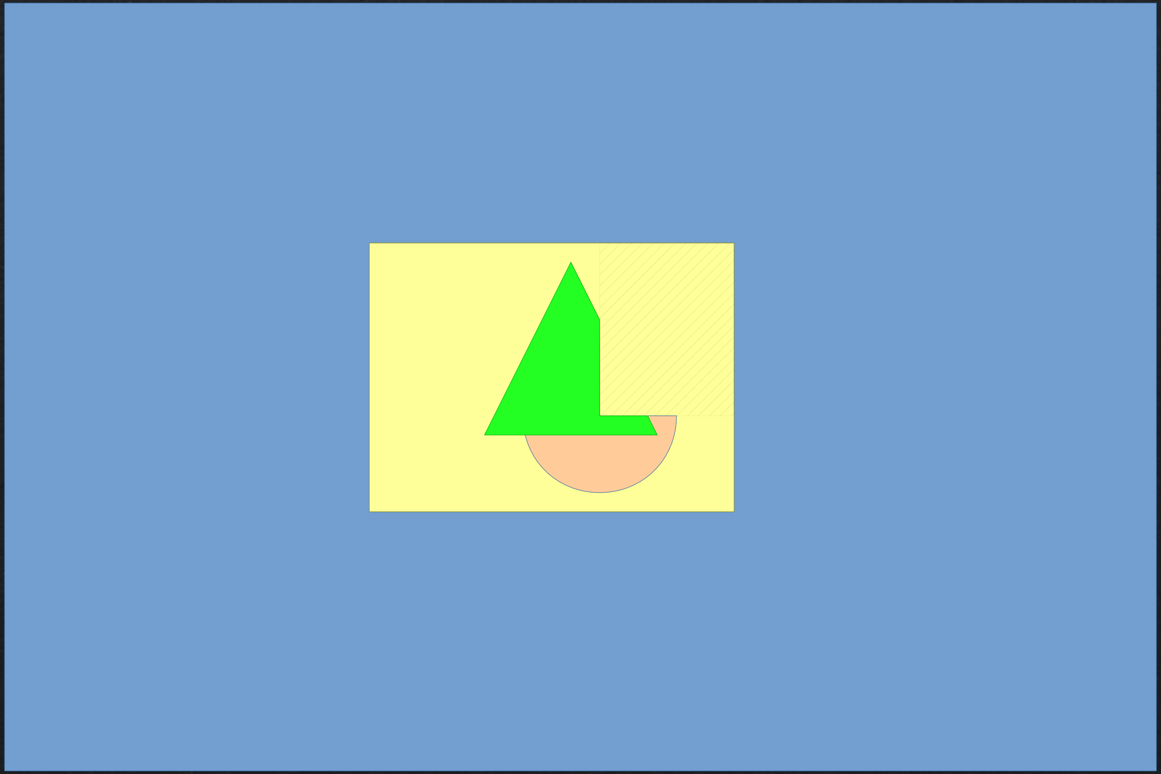
X11 Drawing



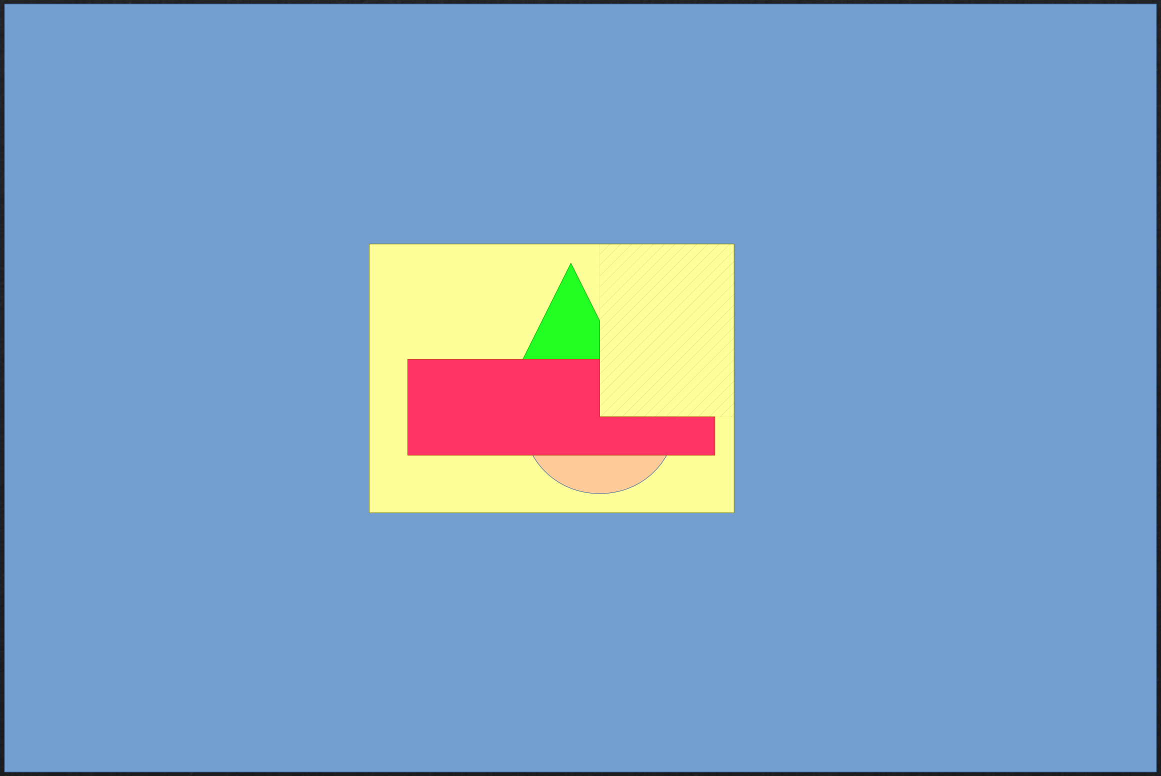
X11 Drawing



X11 Drawing



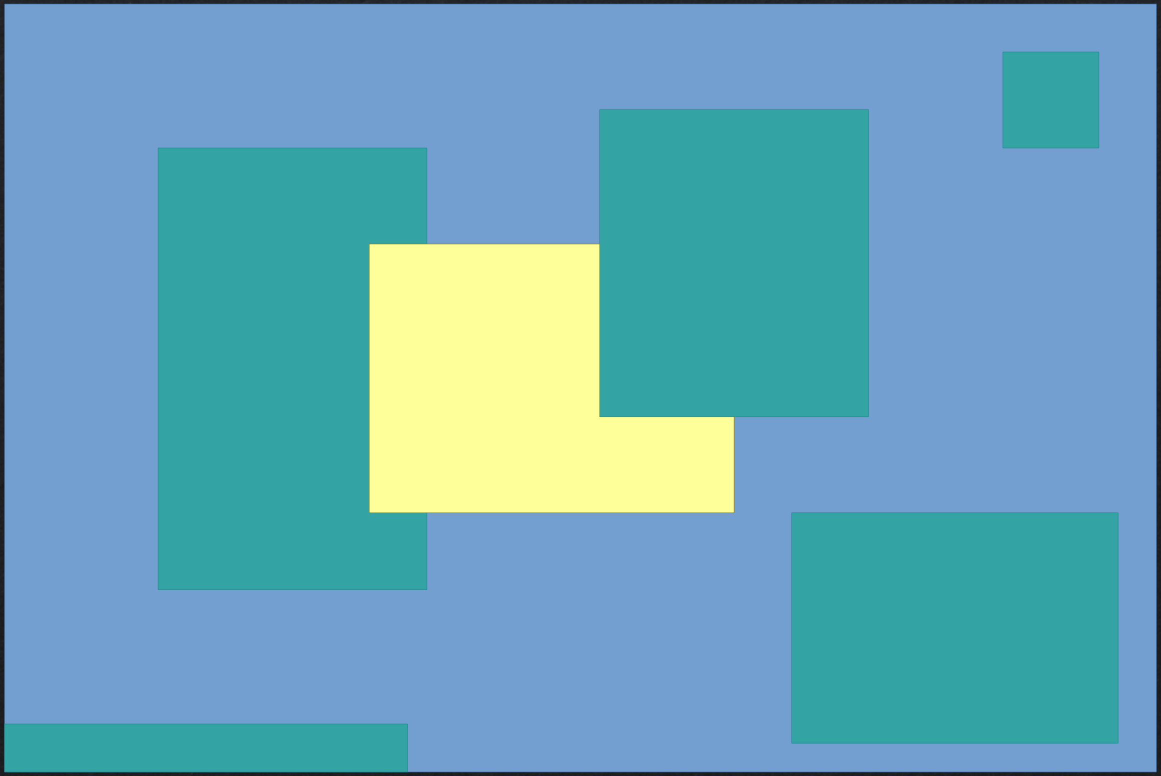
X11 Drawing



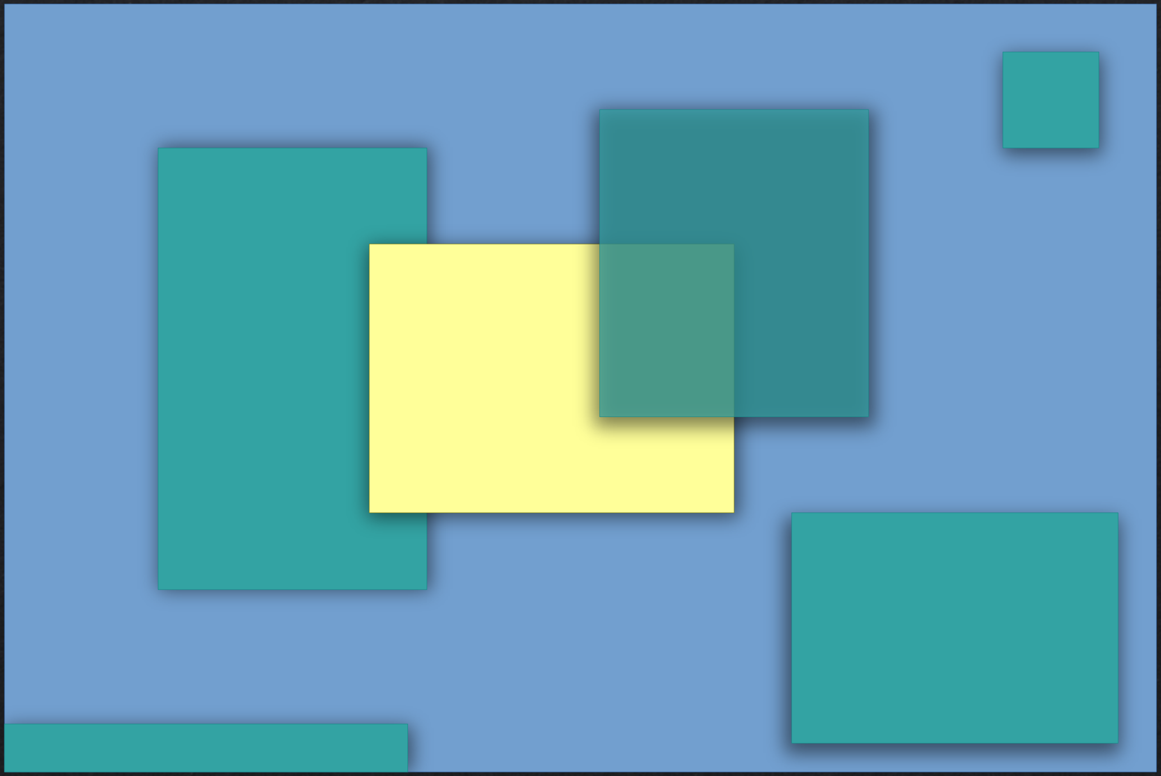
X11 Adds compositing

- Compositing added via several extensions
  - Composite, Damage, Fixes
- Forces renders to go to off-screen pixmap per window
- Allows compositor to get events on changes and pixmap IDs
- This allows compositor to add effects like shadows, zooms
- Downside – can't affect events (go direct to target client)

Compositing



Compositing



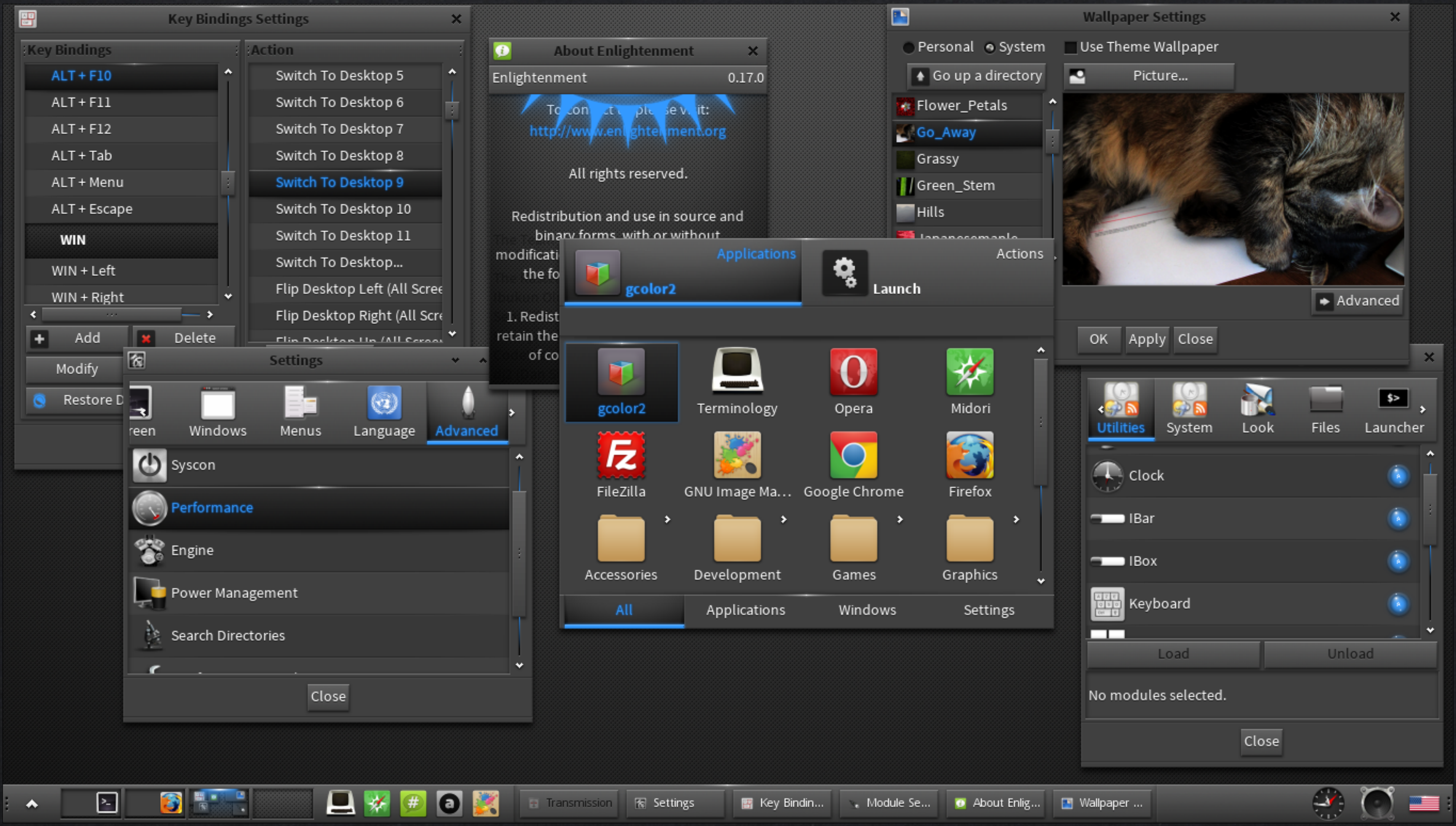
Other OS did the same

- Windows XP to Vista+ added Compositing
- Mac OS9 to OS X added Compositing
- Compositing has the same core ideas across them all
  - All drawing to windows now goes to backing buffer
  - Compositor can access backing buffers & updates
  - Compositor process composes the screen using buffers
  - This composition process can add effects & transparency
- Tizen is composited!

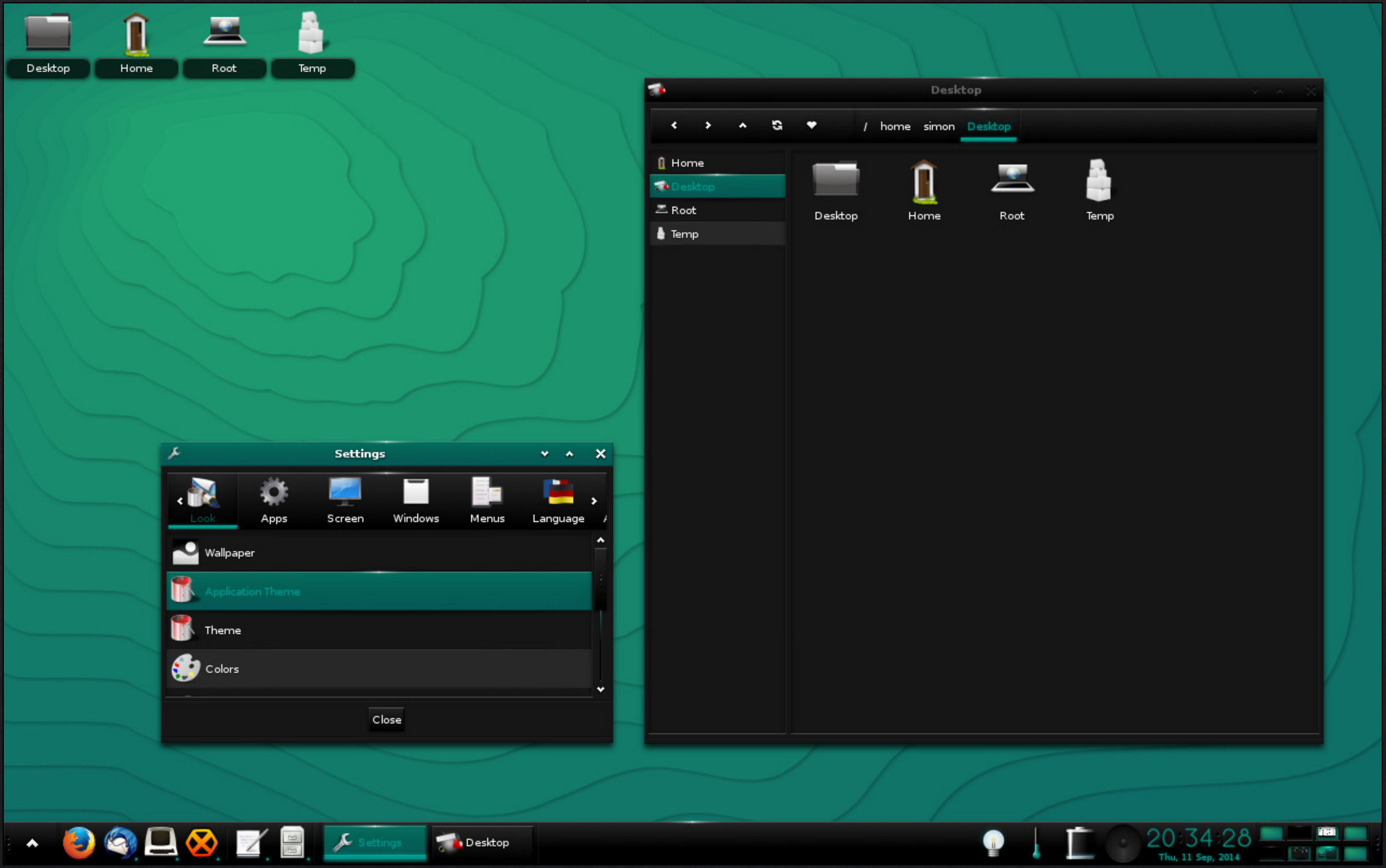
Tizen 2.x Display

- X11 + Compositor (Enlightenment 0.17)
  - Except IVI (Weston/Wayland)
- It is a Full Desktop WM + Compositor underneath
  - Windows can (and will resize)
  - Windows can move around
  - Windows may not fill the screen
  - You can have many windows
- Only some policies (eg mobile) force things to be simpler
  - In most cases resizes don't happen often
  - Windows tend not to move

Tizen 2.x WM


Key Bindings Settings
Key Bindings
ALT + F10
ALT + F11
ALT + F12
ALT + Tab
ALT + Menu
ALT + Escape
WIN
WIN + Left
WIN + Right
Add
Delete
Modify
Restore D
Action
Switch To Desktop 5
Switch To Desktop 6
Switch To Desktop 7
Switch To Desktop 8
Switch To Desktop 9
Switch To Desktop 10
Switch To Desktop 11
Switch To Desktop...
Flip Desktop Left (All Scree
Flip Desktop Right (All Scre
About Enlightenment
Enlightenment 0.17.0
http://www.enlightenment.org
All rights reserved.
Redistribution and use in source and binary forms, with or without
Settings
Windows
Menus
Language
Advanced
Syscon
Performance
Engine
Power Management
Search Directories
Close
Applications
gcolor2
Actions
Launch
Terminology
Opera
Midori
FileZilla
GNU Image Ma...
Google Chrome
Firefox
Accessories
Development
Games
Graphics
All
Applications
Windows
Settings
Wallpaper Settings
Personal
System
Use Theme Wallpaper
Go up a directory
Picture...
Flower_Petals
Go_Away
Grassy
Green_Stem
Hills
Advanced
OK
Apply
Close
Utilities
System
Look
Files
Launcher
Clock
IBar
IBox
Keyboard
Load
Unload
No modules selected.
Close
Transmission
Settings
Key Bindin...
Module Se...
About Enlig...
Wallpaper ...

Tizen 2.x WM


Desktop
Home
Root
Temp
Desktop
/ home simon Desktop
Home
Desktop
Root
Temp
Desktop
Home
Root
Temp
Settings
Look
Apps
Screen
Windows
Menus
Language
Wallpaper
Application Theme
Theme
Colors
Close
Settings
Desktop
20:34:28
Thu, 11 Sep, 2014

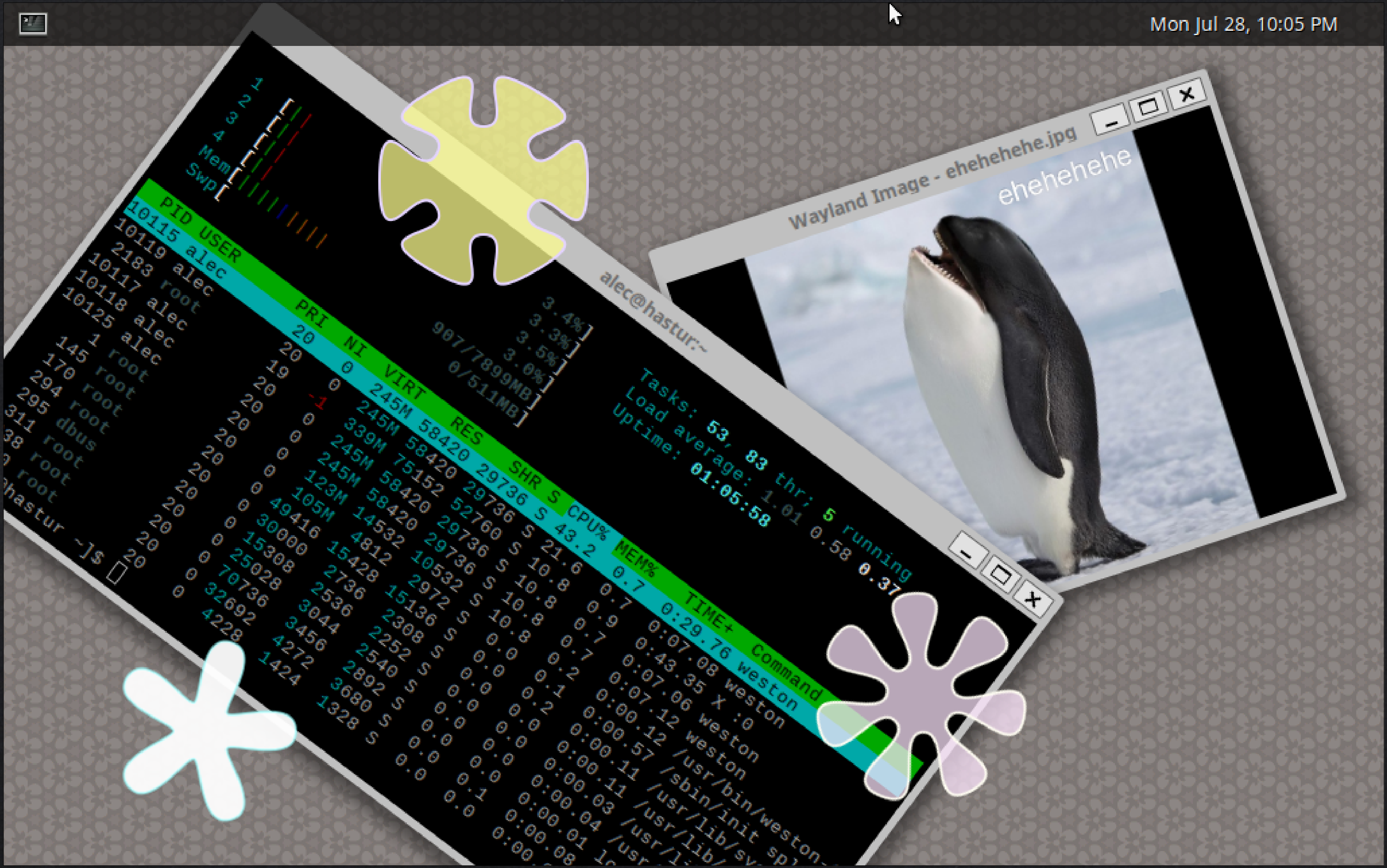
Tizen 3.x Display

- BOTH X11 and Wayland will be supported
  - Moving to Wayland and dropping X11
    - Do not assume / expect or use anything X11
    - Abstractions exist to hide X11 – use them
  - IVI Exception – Wayland ONLY

Wayland

- Far simpler than X11
- Everything is a buffer (or surface) instead of a rectangle
- Composited display ONLY
  - Designed to allow fast-path zero-copy swaps
    - Fullscreen apps (If multiple HW layers, then windows)
- Weston is the current demo compositor
  - More desktops adding support as compositors
    - GNOME, KDE, Enlightenment …
- Major toolkits now have good Wayland support
  - EFL, Qt, GTK+, SDL, ...

Wayland



```
  1  [|||                          3.4%]
  2  [|||                          3.3%]
  3  [|||                          3.5%]
  4  [|||                          3.0%]
  Mem[||||||||             907/7899MB]
  Swp[                        0/511MB]
                                        Tasks: 53, 83 thr; 5 running
                                        Load average: 1.01 0.58 0.37
                                        Uptime: 01:05:58

  PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
10115 alec       20   0  245M 58420 29736 S 43.2  0.7  0:29.76 weston
10119 alec       20   0  339M 75152 52760 S 21.6  0.9  0:07.08 weston
 2183 root       19  -1  245M 58420 29736 S 10.8  0.7  0:43.35 X :0
10117 alec       20   0  123M 14532 10532 S 10.8  0.7  0:07.06 weston
10118 alec       20   0  105M  4812  2972 S 10.8  0.2  0:07.12 weston
10125 alec       20   0 49416 15428 15136 S  0.0  0.1  0:00.12 /usr/bin/weston
    1 root       20   0 30000  2736  2308 S  0.0  0.2  0:00.57 /sbin/init spl
  145 root       20   0 15308  2536  2252 S  0.0  0.0  0:00.11 /usr/lib/sy
  170 root       20   0 25028  3044  2540 S  0.0  0.0  0:00.11 /usr/lib/
  294 root       20   0 70736  3456  2892 S  0.0  0.0  0:00.03 /usr/l
  295 dbus       20   0 32692  4272  3680 S  0.0  0.1  0:00.04 /usr/
  311 root       20   0  4228  1424  1328 S  0.0  0.0  0:00.01 lo
   38 root       20                                    0:00.08
    9 root       20
hastur ~]$ 
```

TIZEN™ DEVELOPER SUMMIT
2014 SHANGHAI
TIZEN开发者峰会（上海）

Why Wayland

- Security possible (X11 is insecure by design)
- Compositor + WM + Display Server in a single process
  - Allows for input transforms easily
  - Far more efficient than multiple processes
  - Lower power consumption
- Compositing model more cleanly allows HW layer usage
  - YUV or RGBA layers can be easily supported (Subsurface)
- Leaner
  - Throws out legacy server-side rendering
  - Clients self-render these days on X11 anyway

Why Wayland

- Input method support is integral, not an afterthough
- DND is integral and not an afterthought
- Provides far more client isolation than X11
- Far less code to support
- Less time to market to bring up new GPUs and boards
- Built around open standards like DRM, KMS etc.
- Much better chances to ensure “every frame is perfect”
- Still client-server for commands + signaling (UNIX socket)
  - Buffers are zero-copy (only handles sent via IPC)

Application Toolkits

Building apps for 1st parties

- Can access any layer (X11, Wayland, FB etc.)
  - Keep in mind portability and moving from X11 to Wayland
- Can use OpenGL directly
- Can use EFL Directly
- Can use Qt Directly
- Etc.

Building apps for 3rd parties

- Use HTML5 + Webruntime
  - Provides HTML5 DOM / CSS / JS
  - Forces most of App to also be in JS
  - Slow startup and heavy memory footprint
  - Performance tradeoffs for development speed/environment

Building apps for 3rd parties

- C++ Tizen::Native API Deprecated

Building apps for 3rd parties

- New Tizen Native C API is here
  - As of Tizen 2.3
  - Many APIs covering all aspects of Tizen Devices
  - UI API is EFL
    - EFL 1.7 + Patches (some EFL libs only)
- C Language as core support
  - Of course C++ works as well

EFL

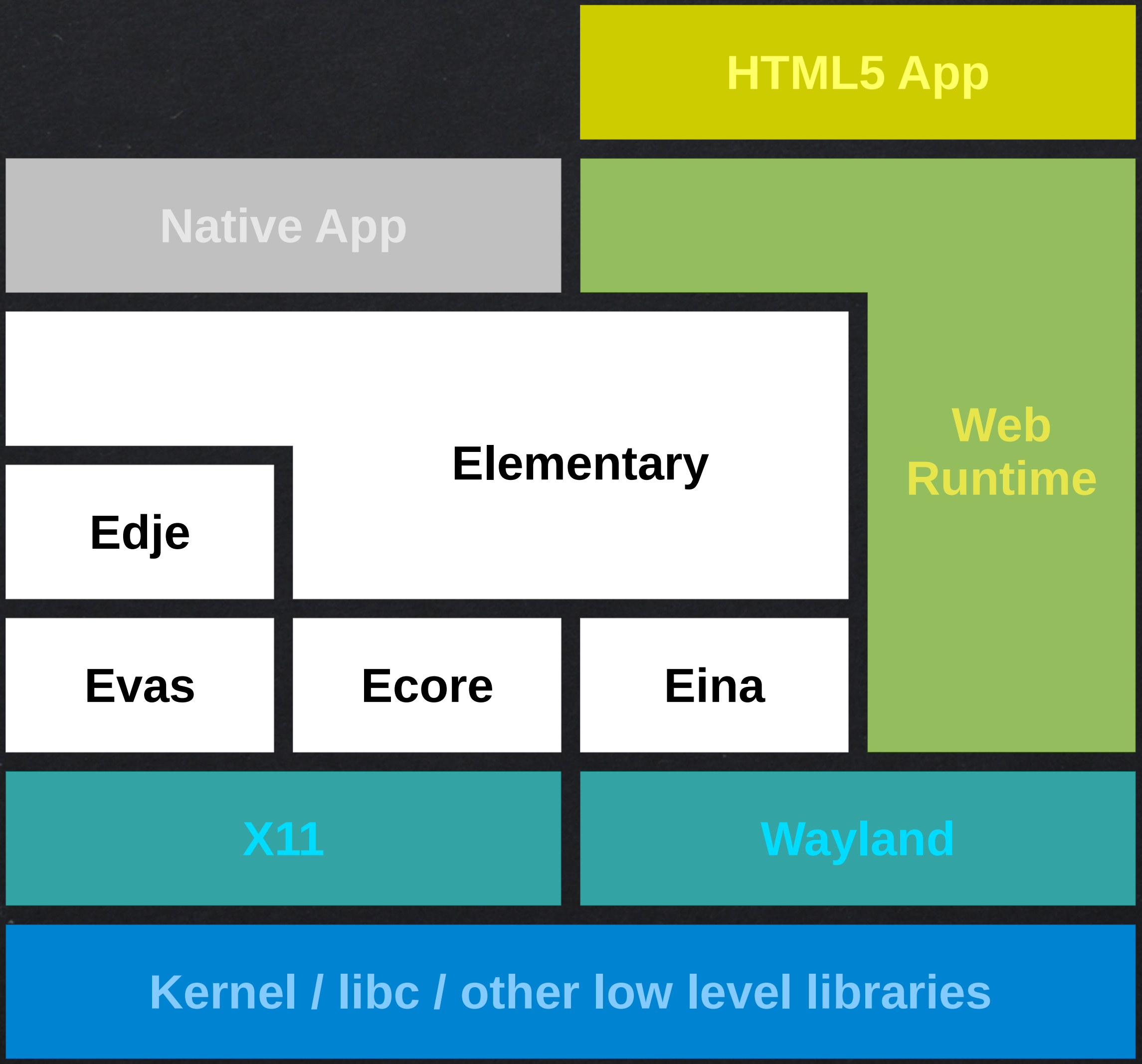
What is it?

- Stands for: Enlightenment Foundation Libraries
- Was created as part of building Enlightenment
  - http://www.enlightenment.org
  - “We need these things and nothing else provides them”
  - “Maybe they will be useful to others too?”
    - Made them libraries instead of compiled-in code
- Were built keeping “Embedded” in mind
- Created & maintained by a very small and focused team
- 100% Open source (development model, community & code)

What is it?

- Today upstream is about 1,000,000 lines of C (1.11)
- Is a C-centric library with C APIs
- Contains sub-libraries with specific names, functions & layers
  - Elementary – High level API + Widgets
  - Evas – Core scene graph + rendering
  - Ecore – Mainloop, events, messaging & animation / timing
  - Eina – Data structures and Low level
  - Edje – Canvas object "meta" files from on-disk themes
  - … and others

Blocks!

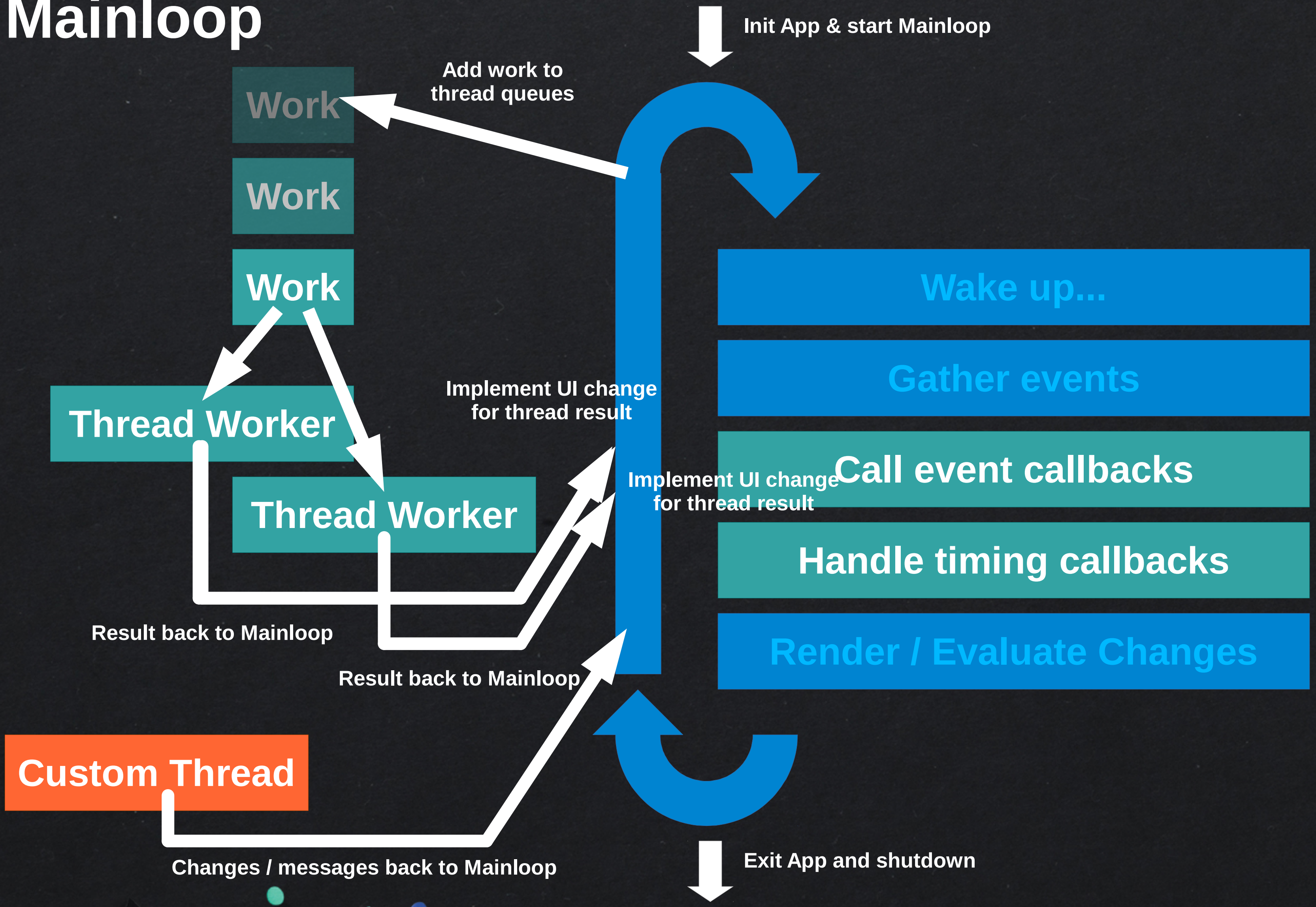


* Rough Block Diagram
NOT LITERAL

Mainloop Model

- EFL is designed to be Mainloop-centric
- Mainloop handles application state and UI in ONE THREAD
- Child threads can do work async and message mainloop
- Thread worker pool APIs provided for you
- Encourages thread isolation of data / tasks
- Encourages serialisation of current state into Mainloop
  - Implicit synchronisation of state changes
  - Fewer locks needed
  - Fewer thread bugs

Mainloop


Init App & start Mainloop
Add work to thread queues
Work
Work
Work
Thread Worker
Thread Worker
Implement UI change for thread result
Implement UI change for thread result
Wake up...
Gather events
Call event callbacks
Handle timing callbacks
Render / Evaluate Changes
Result back to Mainloop
Result back to Mainloop
Custom Thread
Changes / messages back to Mainloop
Exit App and shutdown

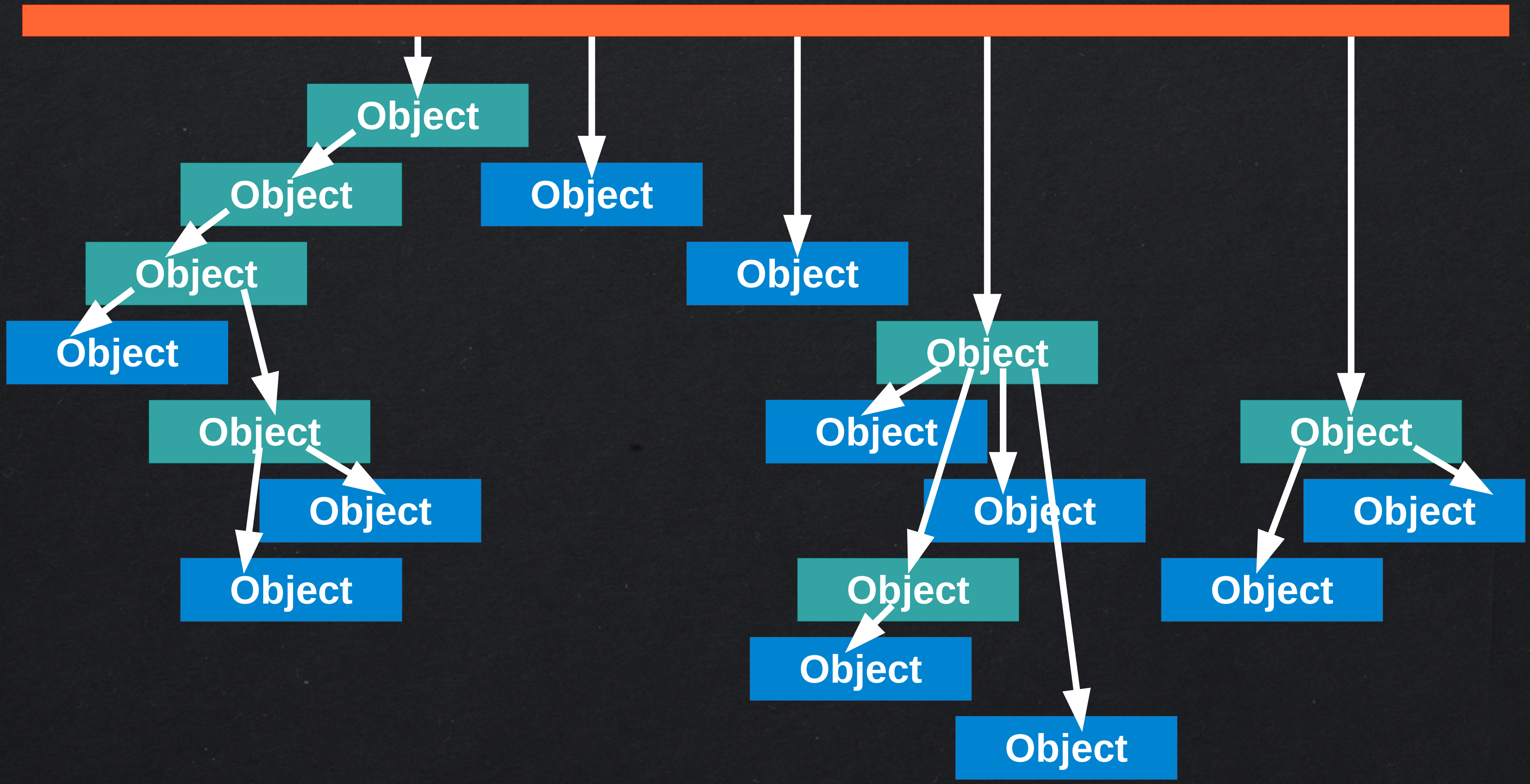
Widgets

- Buttons
- Scrollers
- Entries
- Check buttons
- Radio buttons
- Lists
- Boxes, Tables, Grids
- Menus
- Toolbars
- … and much more

Scene Graph

- Unlike almost everywhere else, there is no “rendering” API
- But there is the SCENE GRAPH (Retained Mode Rendering)
- A scene graph describes your window content by objects
  - Every object is a primitive
    - Text, Image, Rectangle, Container, …
    - Buttons, Tables, Boxes, Lists, ...
  - Do not redraw. Modify objects to achieve changed scene
  - Scene graph figures out how to redraw – not application
  - Scene graph figures out what changed to minimize work
  - More objects == more cost to figure out changes :(

Scene Graph


Object
Object
Object
Object
Object
Object
Object
Object
Object
Object
Object
Object
Object
Object
Object
Object
Object
Object

Scene Graph

- Every object has geometry
- Every object has stacking (above/below a sibling)
- Some objects (Smart objects – Containers) can have children
- Every object that is a toplevel (no parent object) is in a Layer
- Layers have absolute stacking priority (0 below 1, 1 below 2)
- This allows not just spatial arrangement but also Z order
- Z order (stacking) is key to getting some effects/behaviors
- Due to its nature, objects are composed, not “inherited”

I lied – you can render

- For vectors use Cairo + Evas Image Object
  - Wrap Cairo Surface around Image object pixel data
  - Draw to Cairo Surface (can be done in thread)
    - If doing so in a thread, double-buffer image objects
  - Throw out Cairo Surface
  - Set pixel data back and give evas region updates for image
- You can use the same technique for any custom pixel data
  - Image objects are ARGB8888 pixels

I lied – you can render (OpenGL)

- You can “insert” your OpenGL rendering into the scene graph
- Use Elm GLView widget to save you time
- It enforces some “limits” due to it being a scene graph
  - Must use Evas GL interface and context handling
  - This allows your GL rendering to be zero-copy rendered
    - No overhead vs raw rendering (no copies)
    - Allows for your GL to have alpha and overlay objects
    - Allows canvas objects to overlay your GL rendering
- Makes it easy to add text, widgets, HUD and debug games etc.
- Makes GL portable (GLES2 on all platforms).

The future

- Is already here (EFL 1.11 already out)
  - Tizen is unfortunately behind EFL releases (Tizen 2.x)
  - Tizen 3.x is tracking upstream EFL
- Upstream EFL has lots of improvements and added features
  - EO (object infra) for safety and OO in C
  - Evas 3D objects
  - Filters for Text (and coming – images)
  - Optimizations
  - Cleanups
  - Better Windows / Mac support (porting)...

The future

- Video objects...
- C++ API (auto-generated – currently “unstable”)
- LUA app creation + auto-generated API
- Same Python API (auto-generated)
- And so much more...
- http://www.enlightenment.org
- http://git.enlightenment.org
- http://phab.enlightenment.org

Q&A

Ask me
ANYTHING

TIZEN™

DEVELOPER SUMMIT 2014

SHANGHAI

TIZEN开发者峰会（上海）