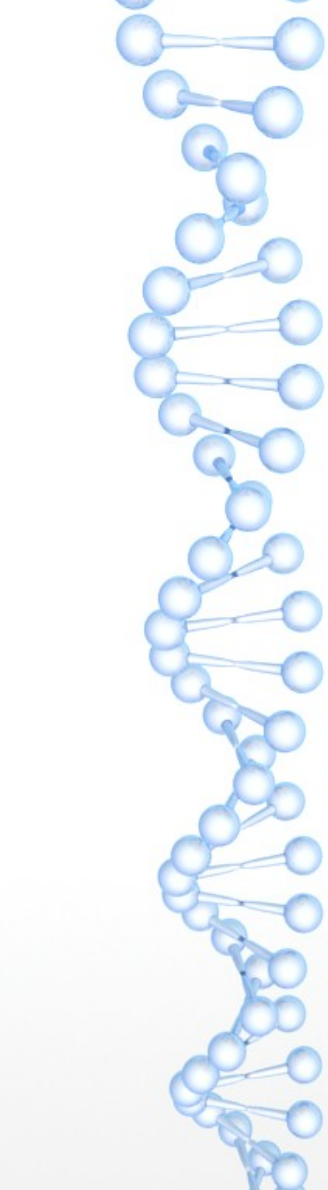
Enlightenment as Standalone Wayland Compositor

Christopher Michael & Stefan Schmidt

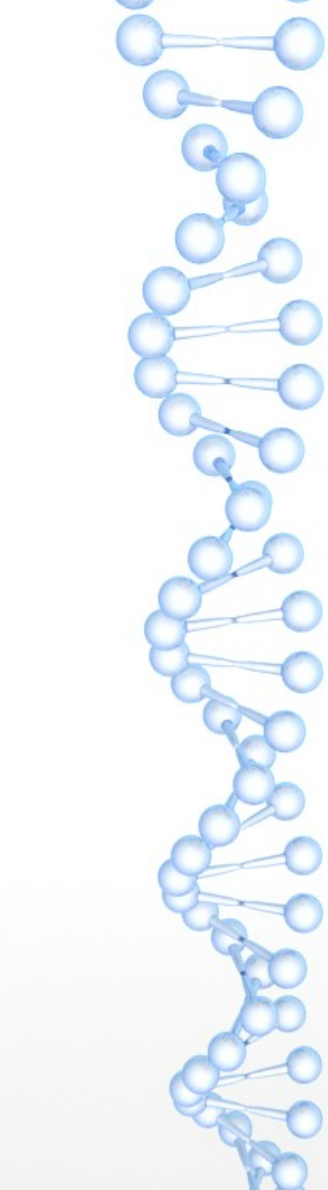
FOSDEM 2014

SAMSUNG
Open Source Group



Intro

- Who are we?
  - EFL upstream developers
  - Working for Samsung Open Source Group UK
  - Direct upstream work

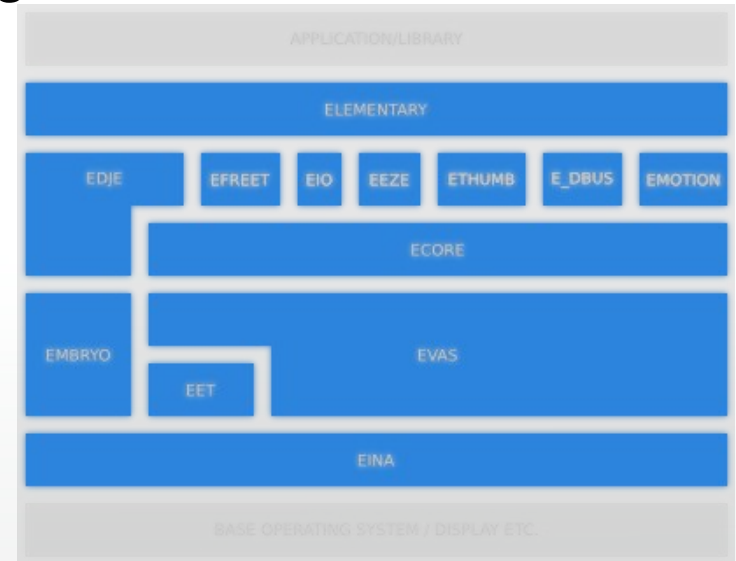


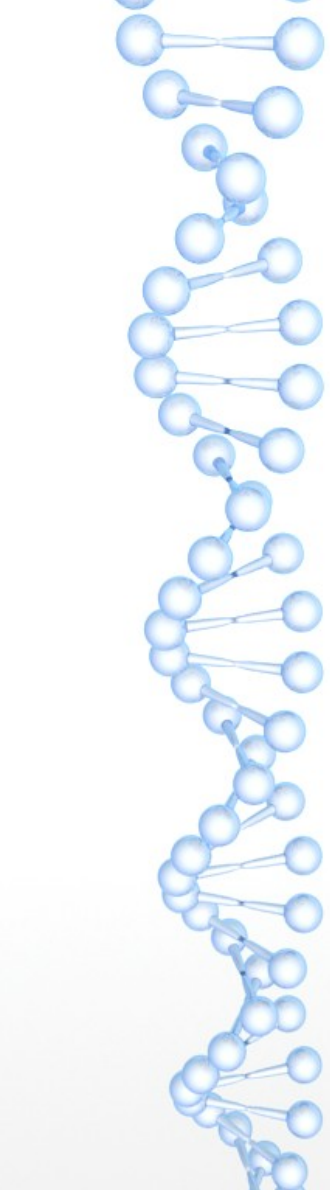
Agenda

- Quick EFL Glossary
- Wayland Toolkit Support in EFL and Elementary
- Step 1: Identify components E relies on from X
- Step 2: Allow Rendering with Wayland
- Step 3: DRM Handling
- Step 4: Input Handling
- Step 5: VT Handling
- Step 6: Session recovery
- Step 7: X Fallback Support through XWayland
- Missing Wayland Parts
- Status
- Summary

Quick EFL Glossary

- Enlightenment Foundation Libraries (EFL): sort of low level libs
- Elementary: a widget toolkit
- Enlightenment: the window manager itself
- Evas: canvas library in EFL
- Ecore_x: our xlib abstraction

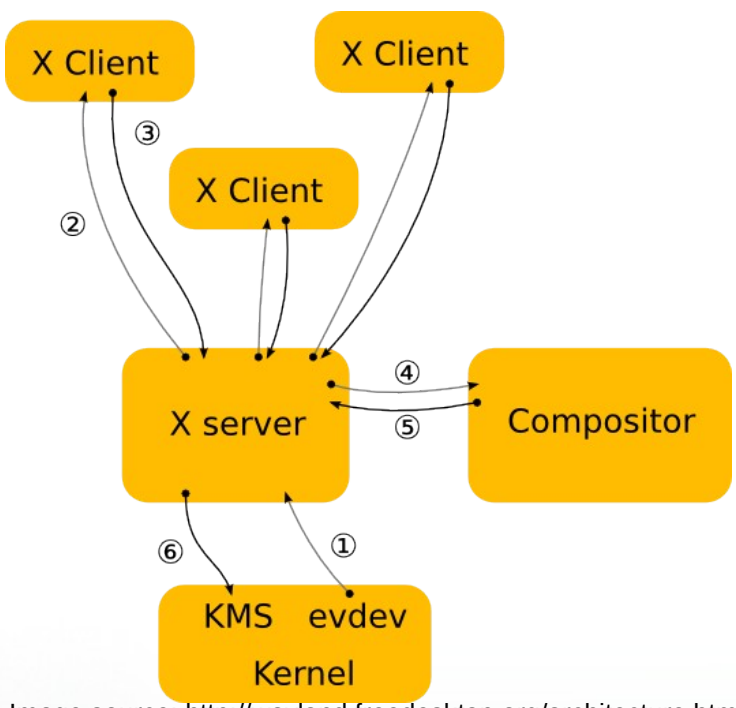




Wayland Toolkit Support in EFL and Elementary

- General Wayland protocol support started by Chris around 2011
- EFL/Elementary Wayland apps running in weston and Enlightenment
- Subsurface protocol
- But this talk is about a standalone wayland compositor :-)

Wayland Architecture



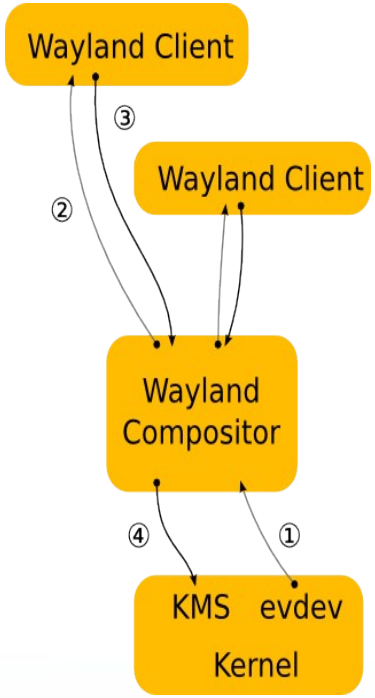
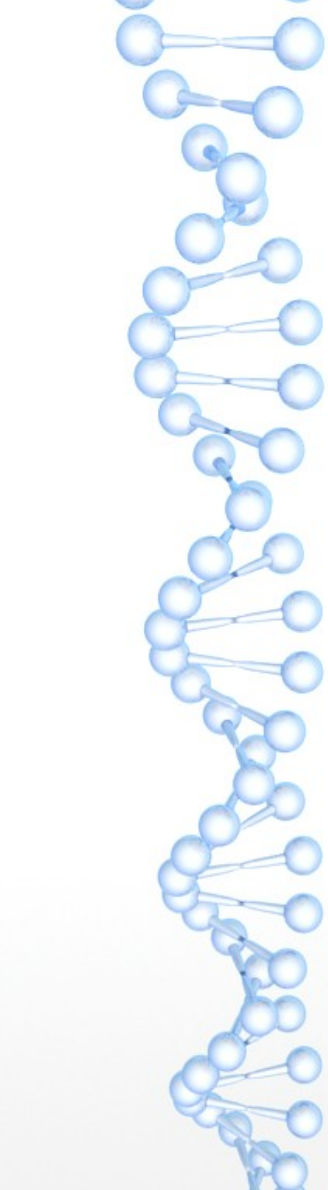
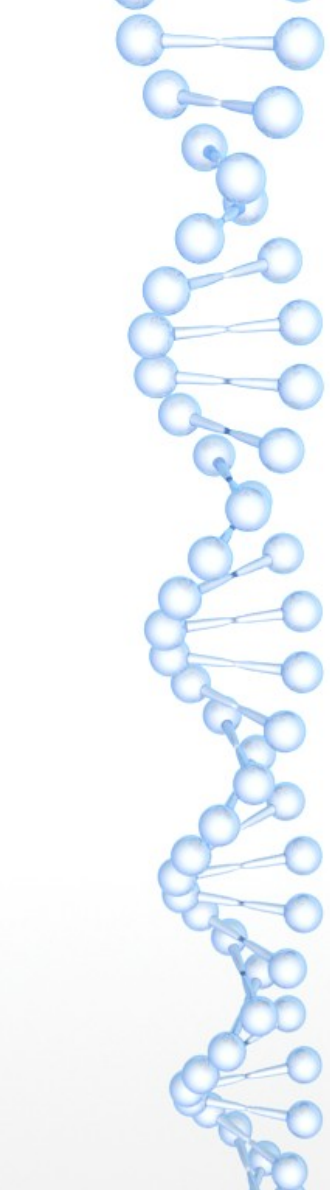


Image source: http://wayland.freedesktop.org/architecture.html



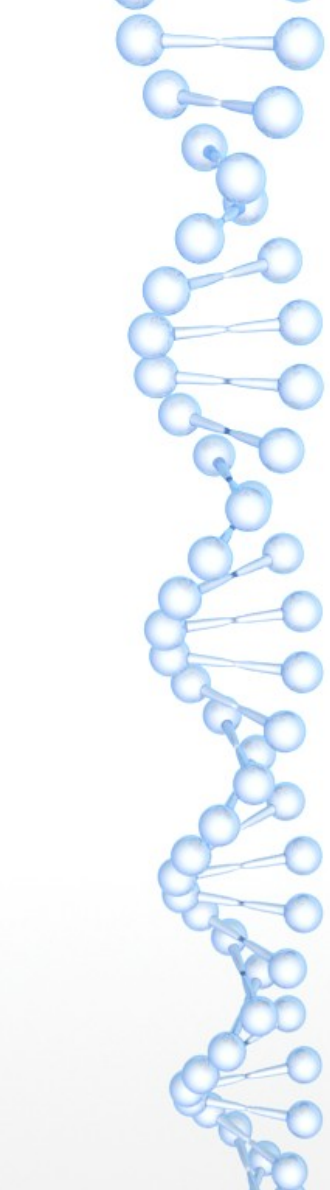
Step 1: Identify components E relies on from X

- Rendering
- DRM handling
- Input handling
- VT handling
- Session recovery



Step 2: Allow Rendering with Wayland

- Wayland engines available in Evas for a long time
  - Shared memory with double and triple buffering
  - EGL engine
- Switched all Xwindow usage to evas canvas to allow X11 as well as wayland surfaces
- Many other abstractions from X already existed in ecore

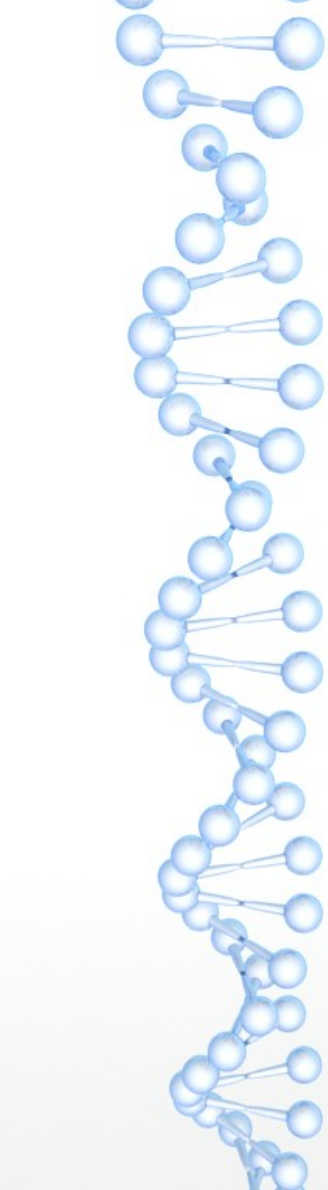


Step 3: DRM Handling (1/2)

- Separate Evas rendering engine
  - Supports software rendering (generic drm FBOs)
  - Supports hardware accelerated rendering (egl)
  - Can be switched Run-Time via Environment Variable
- Not wayland specific (no use of wl_shm buffers or wl_egl windows)
  - Abstracted buffer management
  - GBM (Generic Buffer Management)
  - TBM (Tizen Buffer Management)
  - Others ? (Gem)

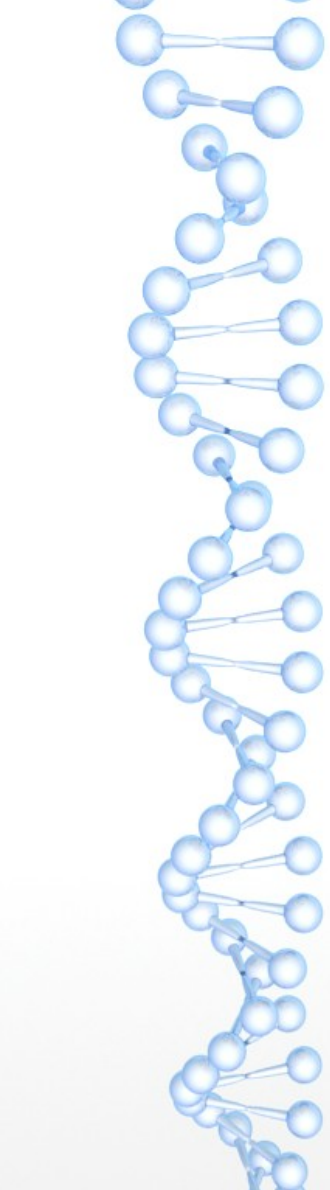
Step 3: DRM Handling (2/2)

- Separate Ecore_Drm library
- Central library for Input, Output, VT Handling
- Implemented using generic drm functions (libdrm)
  - This allows to function via kms or generic fb
- Supports Output Hotplug (via udev)
- Spawns privileged binary for access to restricted input devices
  - Utilizes Unix Socket FD Passing for communication back to main process
- Transparent support for Page Flip & VBlank Events
- Exposes limited API functions
  - Vital (potentially harmful) functions Not exposed to userland



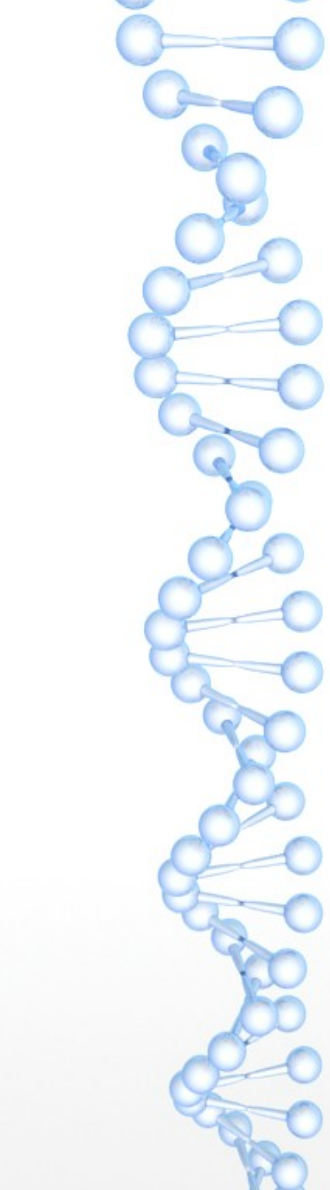
Step 4: Input Handling

- Originally designed to use libinput from Jonas Adahl
  - Removed libinput due to issues with libinput event processing
  - Possibly re-implemented using libinput in the future
- Utilizes Udev for Input Device Discovery
- Supports Evdev devices
  - Keyboard, Mouse, Touchpad, Multi-Touch devices
  - Joystick support currently disabled
- Exposed API functions (via ecore_drm) to dynamically enable/disable input device(s)



Step 5: VT Handling

- Implemented inside ecore_drm library
- Transparent to the user of ecore_drm
- Drops being "drm master" on switch Away from VT
- Acquires "drm master" on switch To the VT
- Uses Proper kernel vt switch signals
  - SIGUSR1 for release
  - SIGUSR2 for acquire



Step 6: Session recovery

- E catches segfault and allows session recovery with all applications restored
- X helps with a lot functionality here
- Wayland protocol has nothing for this yet
- Prototyping something similar as a protocol extension right now

Step 7: X Fallback Support through XWayland

- Wayland protocol support in the major toolkits gets better
- There will be always applications without wayland support (plain X apps, toolkit without wl support, etc)
- We listen on the X socket and start Xwayland on demand
- Starts with the first X client using it and let it time out after the last X client leaves

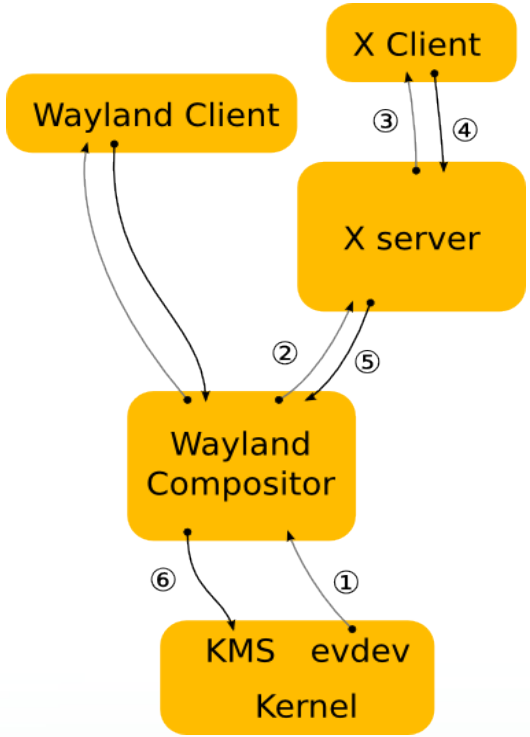
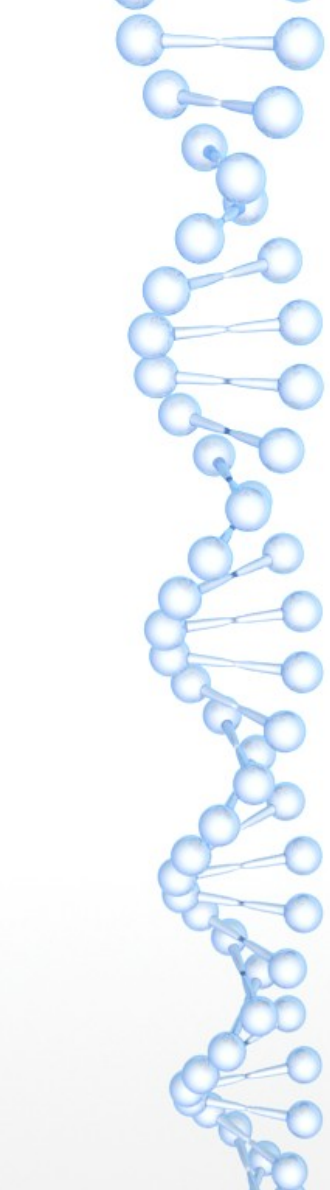
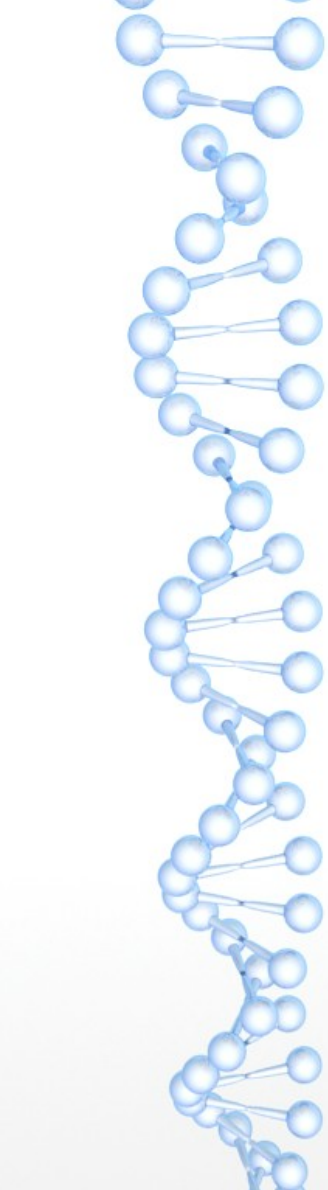


Image source: http://wayland.freedesktop.org/xserver.html



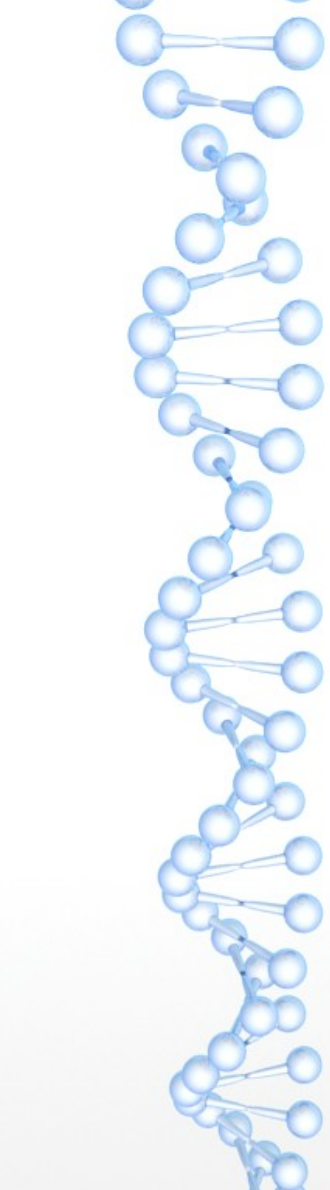
Missing Wayland Parts

- Better support for XDG shell (core protocol is missing desktop related parts, like iconify, systray, border icons, ...)
- But XDG shell in wayland need to mature
- Protocol extension for session recovery
- Feedback from mutter, kwin and Enlightenment helps to identify missing pieces



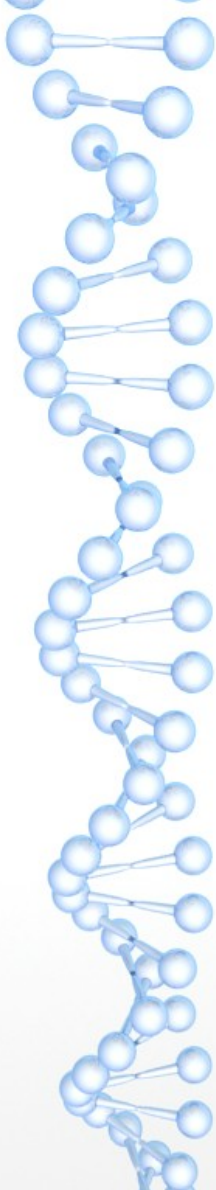
Status

- What do we have working
  - VT switching, input and ouput device handling
  - Running wayland as well as X applications
  - No longer a hard dependency on X in Enlightenment
- What is work in progress
  - Buffer abstraction for rendering
  - Not ready for day to day usage
  - Session recovery



Summary

- Making a X11 window manager act as a standalone Wayland compositor as well is a HUGE task
- The wayland XDG shell extension is missing various pieces to allow for the full desktop experience we are used to
- Things like input handling, VT switching. etc needs to be done by the compositor itself now. Hopefully some sharing between projects.



Thank you!