Wayland Client Basics

Philipp Kerling, Team Kodi
pkerling@casix.org

FOSDEM 2018 Graphics devroom
February 3, 2018

Google Summer of Code

KODI™

Outline

Wayland vs. X: The big picture

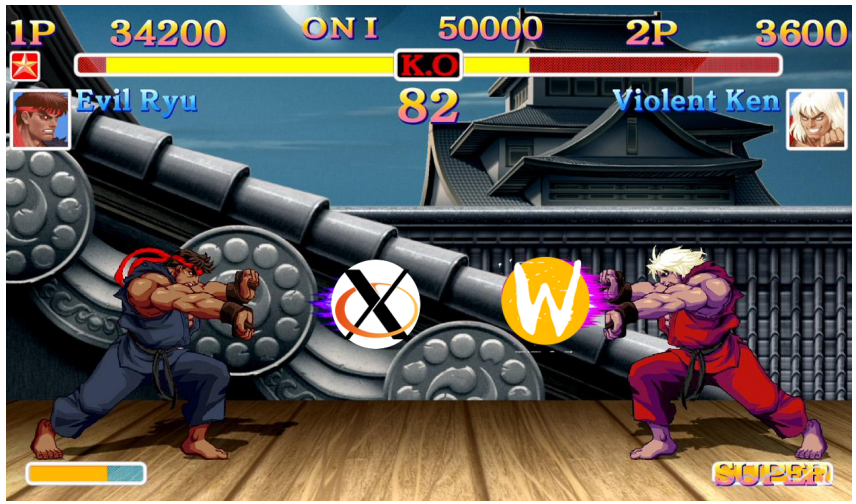
- What is X?
  - X11: Version 11 of the X **protocol**
  - X.Org Server: Canonical server-side implementation of X11 protocol

- What is Wayland?
  - A **protocol** intended to succeed X11
  - Supporting C libraries such as `libwayland-client`
  - Reference server-side implementation `weston`
- What is Wayland not?
  - A piece of software that you can run
  - Like X11

Wayland vs. X: Key differences

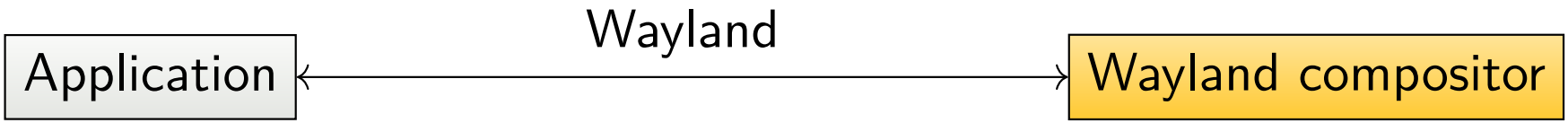
- Wayland *compositors* (servers) combine display server, window manager and compositing functions
- Protocol scope extends beyond traditional desktop environments
- Security-minded design: No **general-purpose interfaces** for e.g.
  - manipulating other windows
  - stealing contents of other windows
  - input injection

  → Instead: **use-case-specific interfaces**
- Less cruft

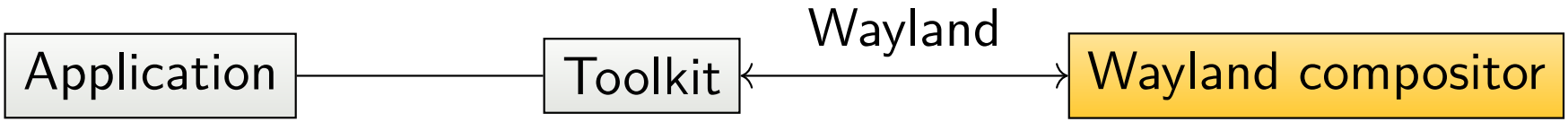


Wayland client implementation options

1. Native Wayland clients (topic of this talk)
   - Lots of effort
   - Beware: Client-side decorations

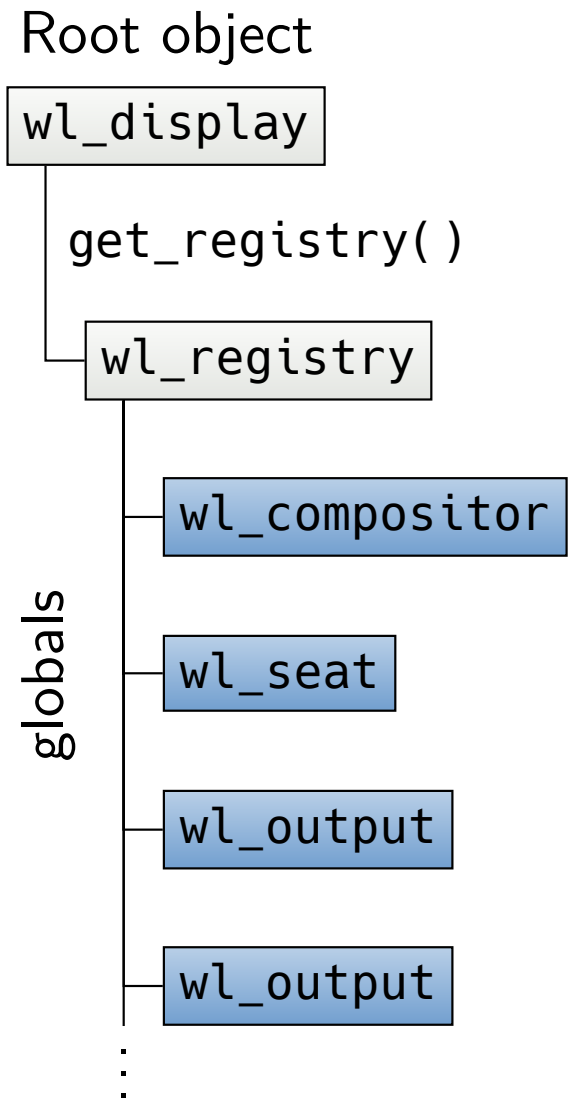


2. **Toolkits!** (Gtk+, Qt, EFL, SDL, GLFW all have Wayland support)



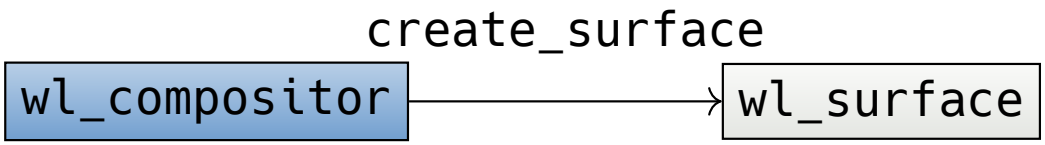
Wayland protocol

- Object-oriented
- XML-based interface definition
  - `wayland.xml`
  - additionally: `wayland-protocols`
- Modular
- Completely asynchronous
- Message-based
- Interface `wl_display`: Singleton representing the connection
- Interface `wl_registry`: Provides access to all *globals*; created from `wl_display`

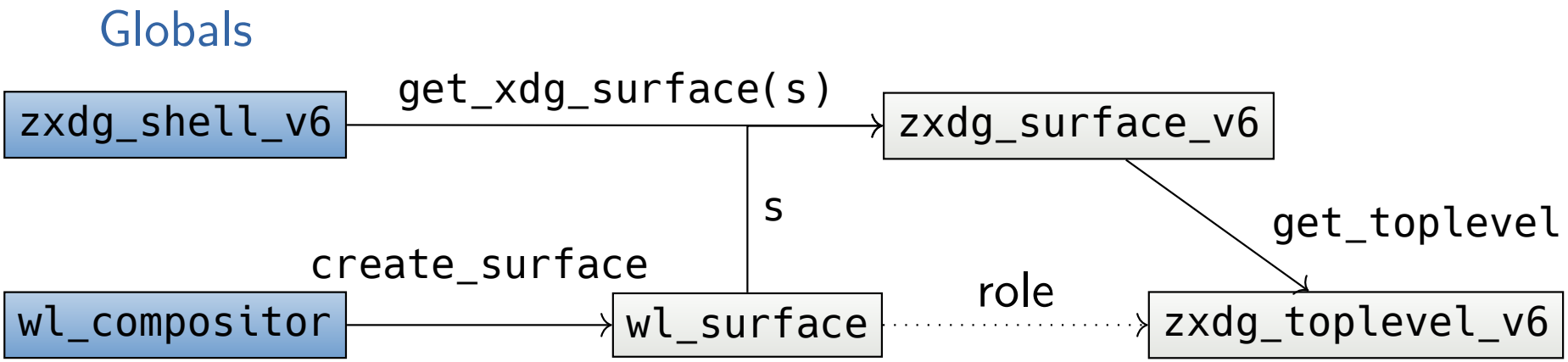


High-level overview of getting a window on screen

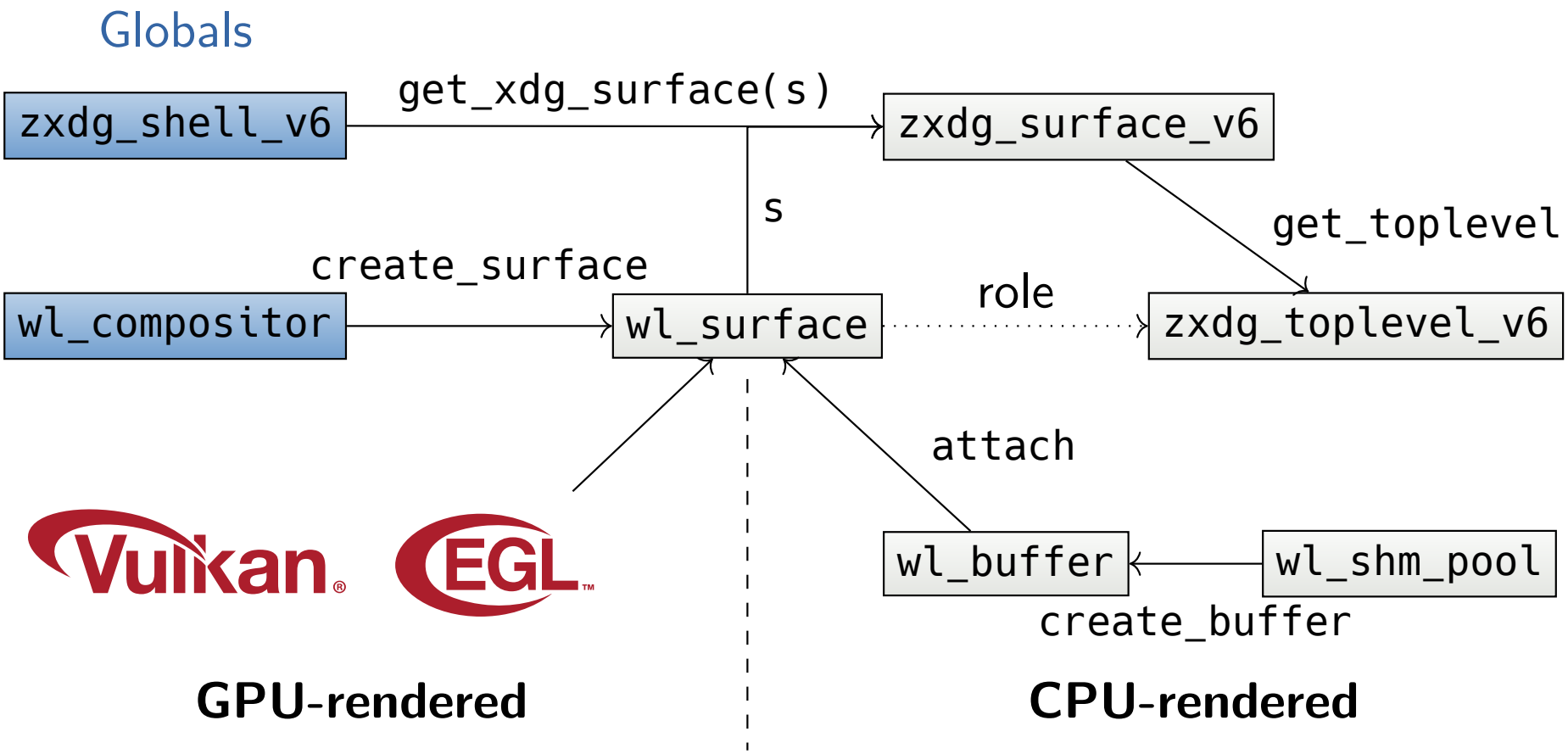
Globals



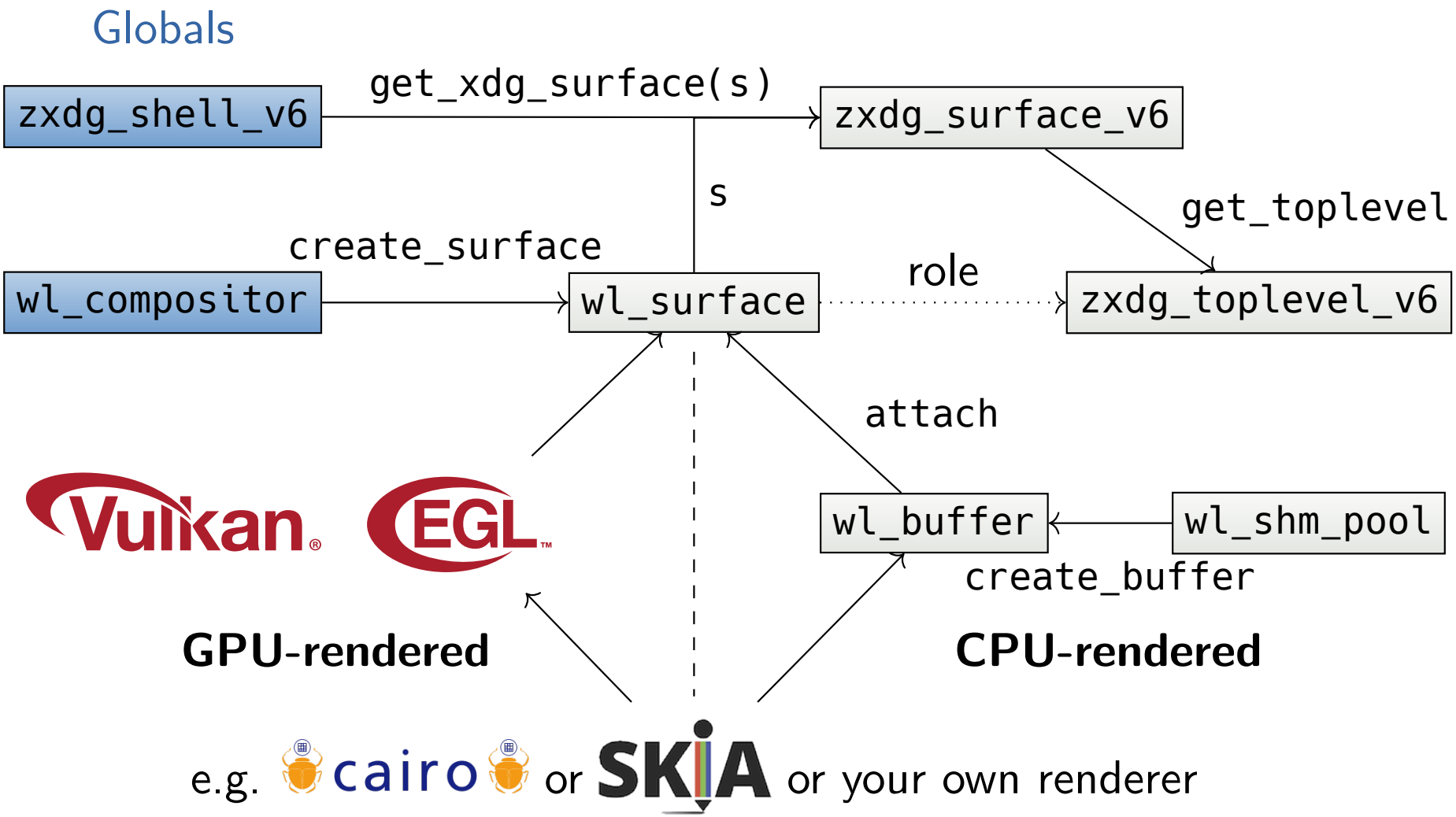
High-level overview of getting a window on screen



High-level overview of getting a window on screen

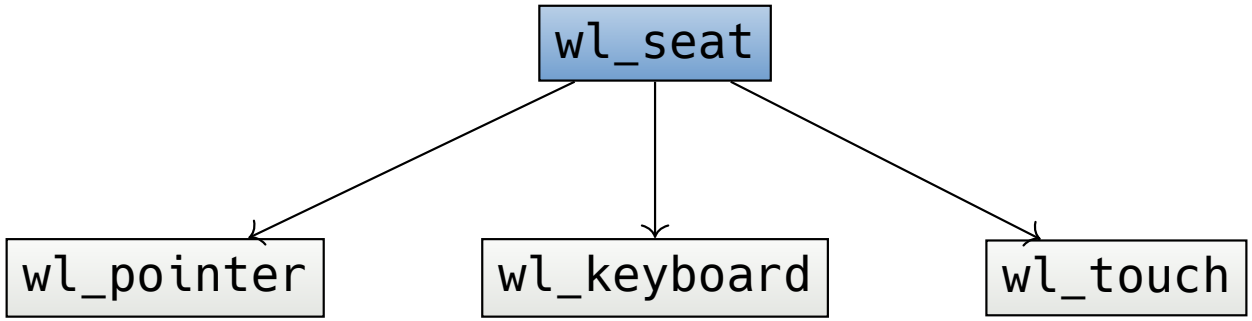


High-level overview of getting a window on screen



Input

- Input devices are grouped in *seats* announced as globals
- Each seat provides access to at most one pointer, keyboard, and/or touch object
- `wl_pointer`: Client must set cursor image when pointer enters surface (`libwayland-cursor` helps with loading Xcursor themes)
- `wl_keyboard`: Server sends keymap that must be parsed with `libxkbcommon`

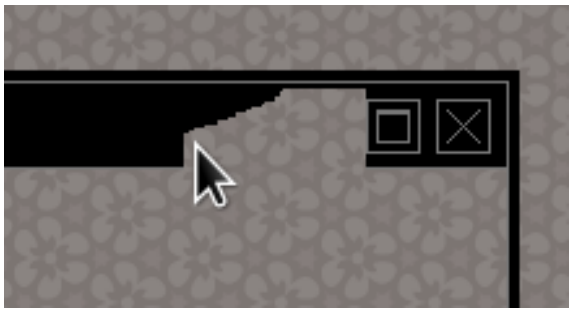


Client protocol implementation

- `libwayland-client` is *the* C library
- Use `wayland-scanner` for any extra protocols
- Start by looking at `simple-shm.c` and/or `simple-egl.c`, part of weston source
- Event loop: Use proper API! (see `wl_display_prepare_read_queue`)
- Be wary of threads
- Bindings: C++ (waylandpp), D (wayland-d), Java (wayland-java-bindings), Rust (wayland-rs)

Testing and debugging

- Run compositors in nested mode:
  - Weston (reference, 3.0.0): `weston`
  - Mutter (GNOME, 3.22): `mutter --wayland --nested`
  - KWin (KDE, 5.12): `kwin_wayland --socket wayland-1 --xwayland`
  - Enlightenment (22): `enlightenment_start`
  - Then: `WAYLAND_DISPLAY=wayland-1 ./my-app`
- `WAYLAND_DEBUG=1 ./my-app` to see all protocol interactions
- Run `weston-info` to see available globals and additional info
- Sometimes, the bug is in the compositor :-)



Conclusion and further resources

- There is a lot more to this...
- Think twice before not using a toolkit
- Project homepage has main protocol and C API documentation
- Extra protocols: documented XML definitions in `wayland-protocols`
- Pekka Paalanen's blog has in-depth reviews of some technical aspects
- For specific questions/problems:
  `wayland-devel@lists.freedesktop.org` or `#wayland` on FreeNode

Thank you for your attention!

Questions?