

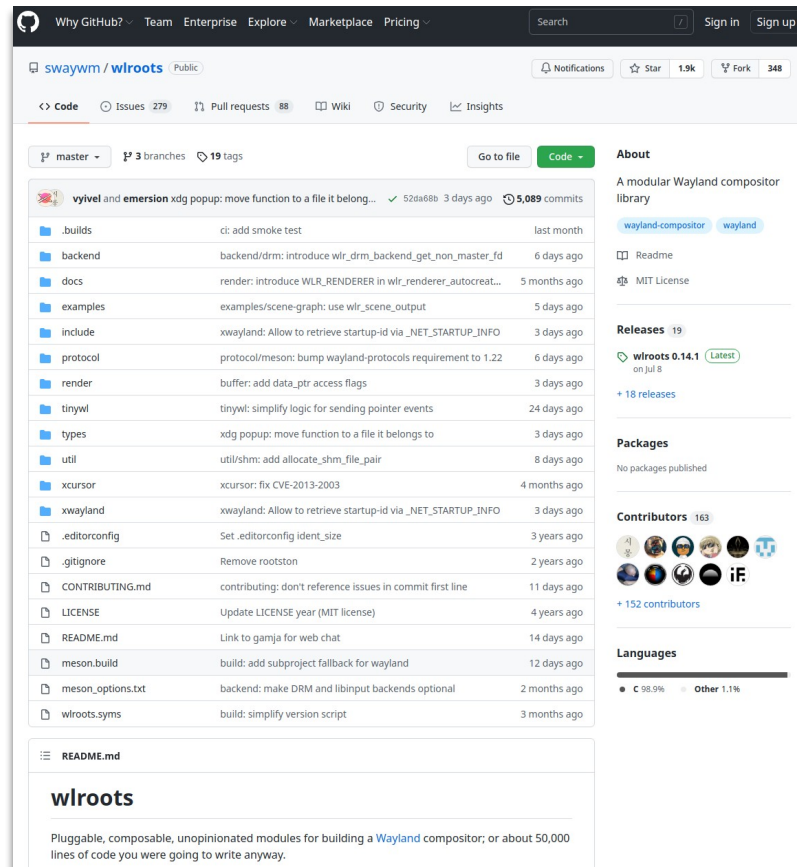


# KWinFT's wroots backend

Roman Gilg

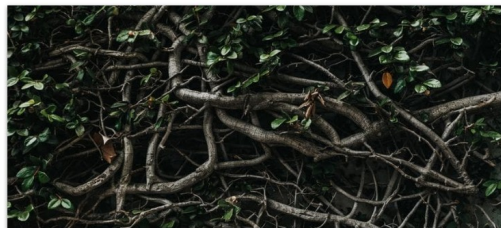
# What is wlroots?

- C library to build Wayland compositors
- Used by several projects:
  - Sway
  - gamescope
  - phoc
  - and now KWinFT
- Actively developed, regular releases



# Why wlroots?

- C library offers easy integration
- Meson build system
- Toolkit approach
- Contributors nice, active in upstream



## wlroots in KWinFT

16. July 2021, by  Roman Gilg — 8 min read.

[kwinft](#) [wlroots](#) [weston](#) [mir](#)

As already teased in the last post about [KWinFT in its second year](#) a few months ago I started with an ambitious endeavour: to replace KWinFT's own platform backends with a single one based on wlroots.

Today the feature branch for this fundamental internal refactoring [has been merged](#) into KWinFT's master branch. To celebrate this milestone let's take a look at the new wlroots backend in more detail.

### 🔗 The wlroots Library

wlroots is a library for Wayland compositor creation. It provides functions and structures to build a Wayland compositor.

Broadly we can divide wlroots into two areas:

- A *server* part, that provides the server-side Wayland functionality; clients talk to it via the Wayland protocol.
- Multiple *backends*, that talk to the platform we run the server on. For example there is a [DRM backend](#) to send pixels to a screen.

wlroots is very well modularized with amenable facilities. It is explicitly not a *midlayer* but a *toolkit* that allows consumers to pick and extend its functionality with ease. In KWinFT we make only use of wlroots' backends.

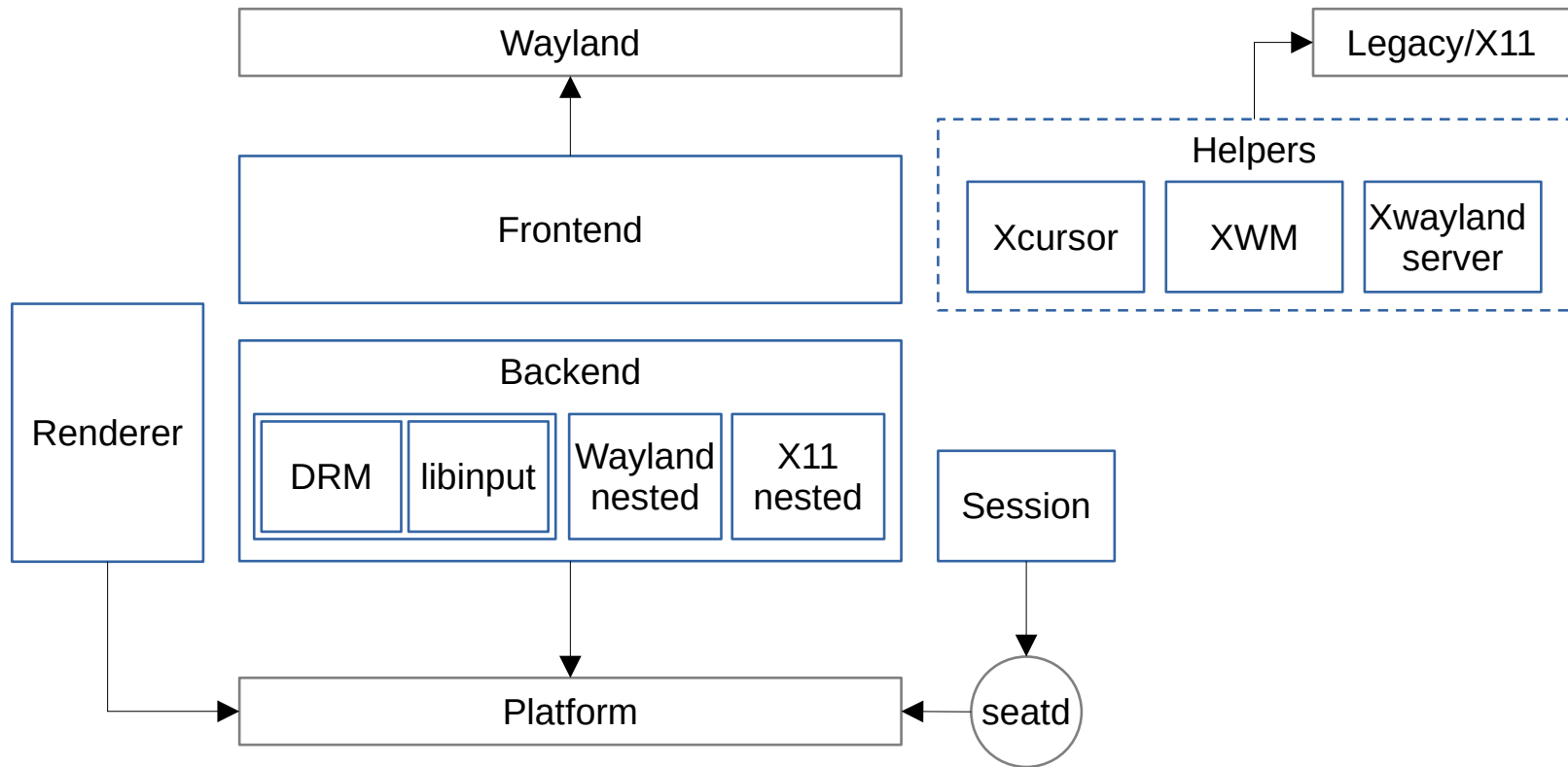
### 🔗 Why Choose wlroots

<https://subdiff.org/blog/2021/wlroots-in-kwinft>

# A wlroots Primer

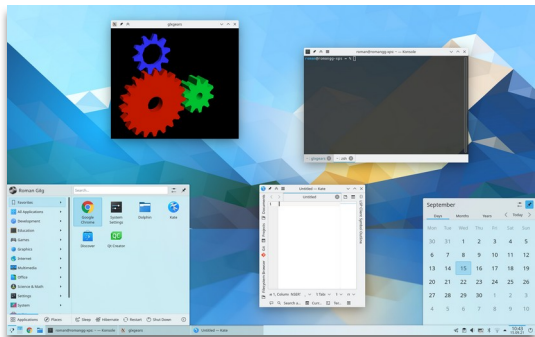
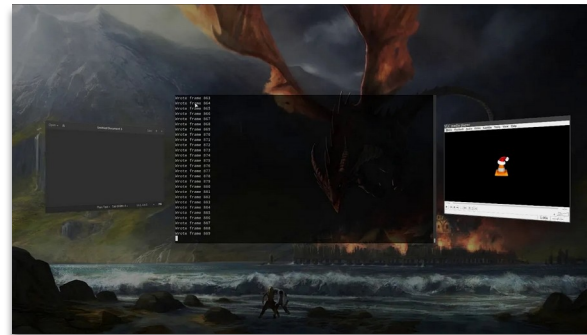
- Split up into a “backend” and a “frontend”
  - Backend allows to communicate with the platform/hardware
  - Frontend provides types for talking the Wayland protocol
- In between but independent: EGL and Pixman renderer
- Also independent:
  - session component (seatd backend)
  - Xcursor library
  - XWM and Xwayland server

# A wlroots Overview



# Consumer Examples

- Only Backend: KWinFT
- Only Frontend: gamescope
- Using both: Sway, Wayfire






# Consumer Quick Start

- Backend:
  - struct wlr\_backend
  - wlr\_backend\_autocreate(wl\_display)
- “Common” types:
  - struct wlr\_output
  - struct wlr\_keyboard
- Frontend:
  - struct wlr\_seat
  - wlr\_seat\_create(wl\_display, ..)

wlroots / include / wlr /

 agx and emersion ...	✓ 3 days ago
..	
backend	6 days ago
interfaces	2 months ago
render	2 months ago
types	3 days ago
util	21 days ago
backend.h	2 months ago
config.h.in	2 months ago
meson.build	7 months ago
version.h.in	3 months ago
xcursor.h	2 months ago
xwayland.h	3 days ago




# In KWinFT


- For now only using the backend part:
  - Session integration
  - Graphical output (via dmabuf)
  - Input processing
- Using our internal renderer
- Goal: Making use of wlroot's renderer
  - Direct scanout
  - libliftoff
  - Future improvements



# Planning

- Started with an issue ticket
- First only render backend
- Feedback from Simon

 KWinFT > KWinFT > Issues > #137

Open Created 6 months ago by  Roman Gilg

## Render and Display backends based on wlroots

### Motivation

We currently implement several display backends ourselves for our Wayland session. These "Platform" called [plugins](#) are:

- DRM
- fbdev
- hwcomposer
- Virtual
- Wayland nested
- X11 nested

The Wayland compositor library *wlroots* [provides](#) the relevant ones of these as backend components too. The idea is to replace our own implementations with the wlroots backend components.

### Advantages:

- Removes a lot of code directly talking with kernel, X11 and other Wayland servers as client.
- Profit from already implemented and upcoming features implemented upstream in wlroots, for example VRR now and in the future libliftoff.
- More shared code with other compositor projects based on wlroots, less fragmentation in Wayland ecosystem.

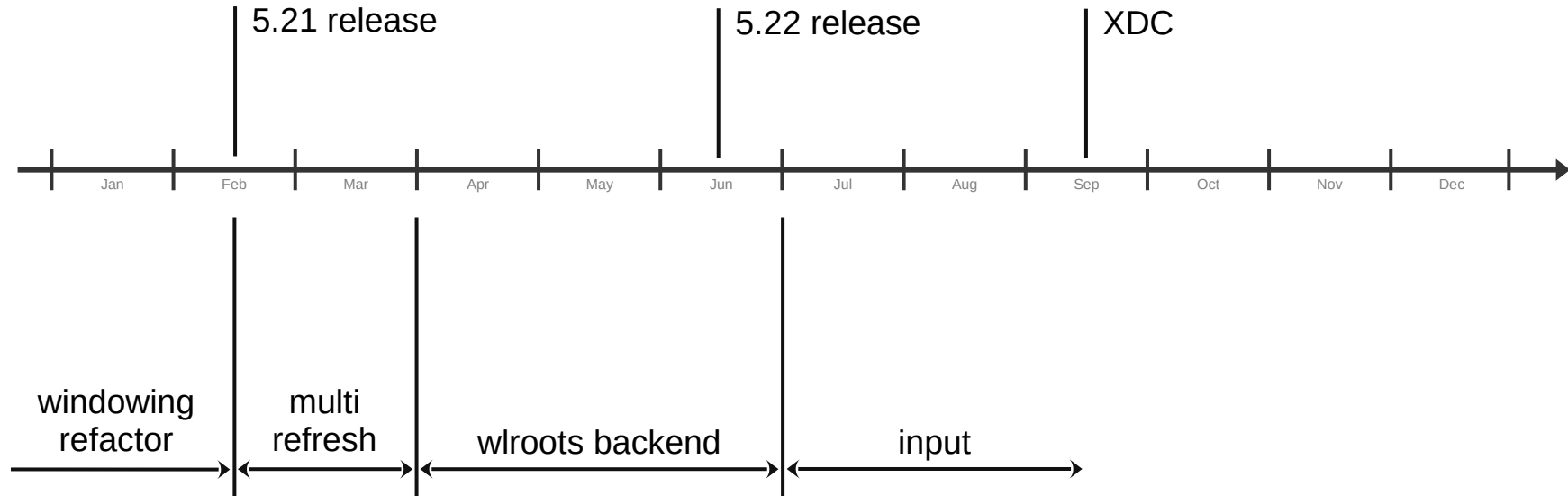
### Early Planning

While the display backends itself are well encapsulated as the Platform plugins, there is integration with the Scene plugins as part of the render pipeline. Additionally there is some integration with logind for the DRM plugin (to become the DRM master).

# Implementation

- Work on separate branch
- Minimal vertical slice
- Needed session and input through wlroots too
  - Removing own libinput backend
  - Own logind backend abstracted
  - Refactoring input and session code
- Make basics work: session, input, output, multi display
- Remove all our previous platform plugins

# KWinFT's Second Year 2021





# The Finish

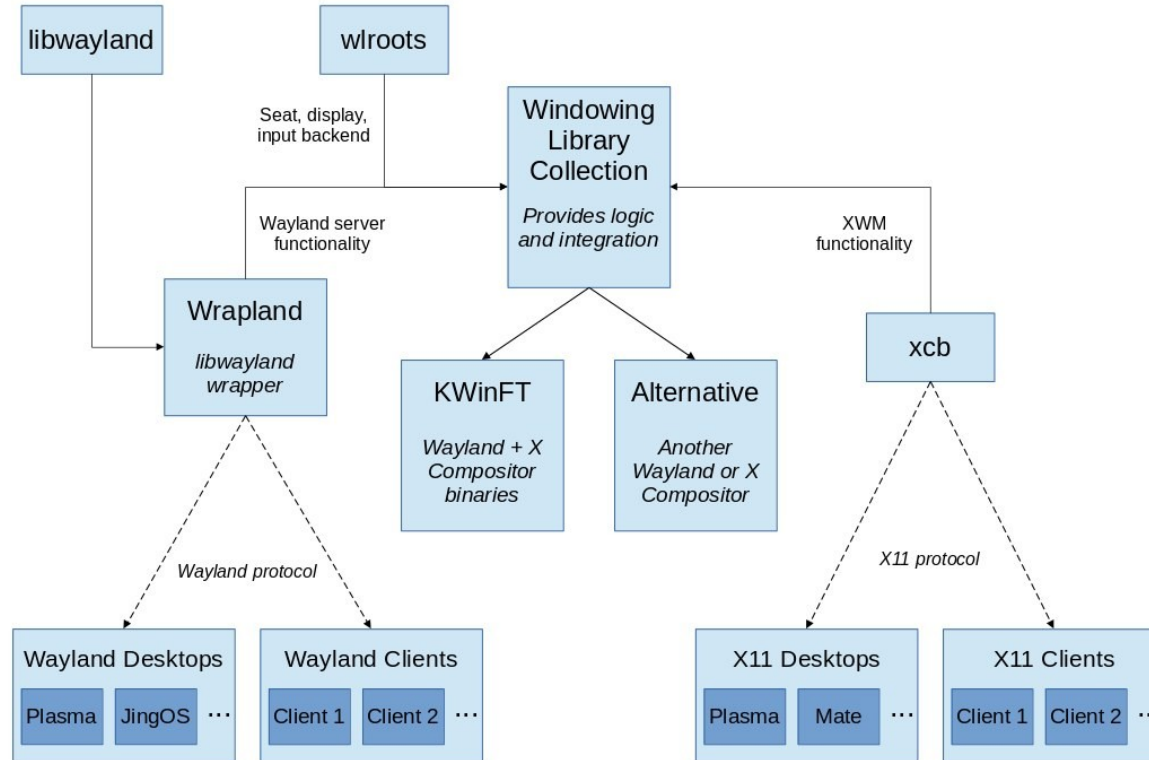
- More testing on master branch → Fixes for standby, pointer gestures
- Simplify startup logic



# Advantages

- Integration tests use same code paths as live session
- Slimmed down code base
  - decreases maintaining cost
  - reduces complexity
- Working together instead of apart
  - more efficient
  - more communication

# wlroots in the Future Ecosystem

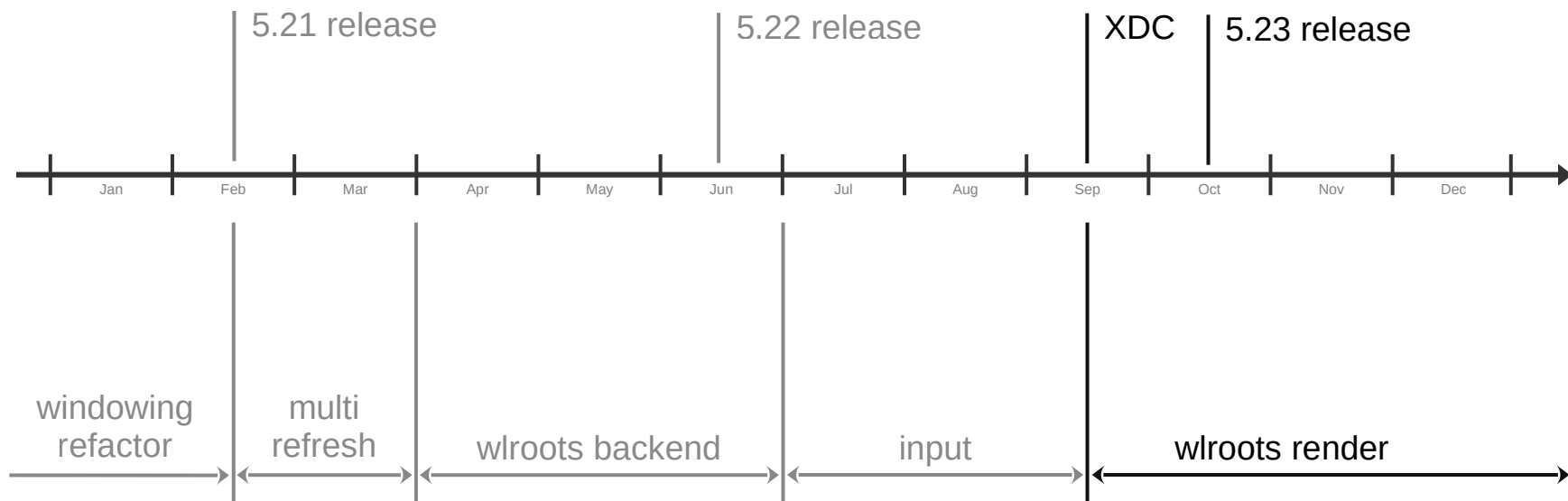


# Next Step: Render With wlroots

- Currently KWinFT uses old render code, result exported to wlroots backend as dmabuf
- Goal is to use the wlroots renderer instead
- More synergies, making use of future libliftoff integration in wlroots
- Challenges:
  - Old renderer uses desktop OpenGL, wlroots only OpenGL ES
  - wlroots renderer code currently moving target
- Require good planning and communication with upstream



# Milestones for 2021



# Milestones for 2022

