Wayland and Weston: 8 years of production devices

Daniel Stone
Graphics Lead, Collabora Ltd.
Hi, I’m Dan

Graphics lead at Collabora
Wayland developer since 2012
daniels@collabora.com
Hi, Dan! What are you talking about?

- This is not about the technical details of Wayland/Weston
  - ... well, not entirely
- The first scratch commit of Wayland was in 2008
  - It was challenging X11, DirectFB, and others
Hi, Dan! What are you talking about?

- This is not about the technical details of Wayland/Weston
  - ... well, not entirely
- The first scratch commit of Wayland was in 2008
  - It was challenging X11, DirectFB, and others
- In 2020, Wayland is now the window system of choice
  - We made a lot of good choices along the way!
Hi, Dan! What are you talking about?

- This is not about the technical details of Wayland/Weston
  - ... well, not entirely
- The first scratch commit of Wayland was in 2008
  - It was challenging X11, DirectFB, and others
- In 2020, Wayland is now the window system of choice
  - We made a lot of good choices along the way
  - And some bad ones too :(
Hi, Dan! What are you talking about?

• This is not about the technical details of Wayland/Weston
  - ... well, not entirely
• The first scratch commit of Wayland was in 2008
  - It was challenging X11, DirectFB, and others
• In 2020, Wayland is now the window system of choice
  - We made a lot of good choices along the way
  - And some bad ones too :( 
• Can you learn anything from us?
Recap: a quick history lesson
Wayland timeline: pre-history

- **Sep 2008** First commit
  - ‘A nano display server’

- **Aug 2010** First protocol
  - Protocol generated from XML

- **Aug 2010** First desktop
  - `wl_shell` and `wl_buffer` added

- **Oct 2008** First headline
  - Phoronix spots the Git repository

- **Dec 2008** First contributor
  - Ray Strode fixes RGB/BGR swap

- **Oct 2010** First post
  - Mailing list on freedesktop.org

**COLLABORA**

Open First
Wayland timeline: Stone Age

- **Feb 2011**: First dependency
  - Wayland and Weston repositories split

- **Mar 2011**: First product
  - Freebox ships to millions of users

- **Feb 2012**: First release
  - Wayland 0.85 released

- **Nov 2010**: First inception
  - Nested Wayland backend added

- **Jun 2011**: First legacy
  - Xwayland hosts X11 clients in Wayland

- **Feb 2012**: First conference
  - 15 contributors meet pre-FOSDEM
Wayland timeline: Feudalism

- **Dec 2013**: Desktop protocols, xdg-shell introduced to replace wl_shell
- **Nov 2013**: First phone, SailfishOS-based Jolla phone ships
- **Apr 2014**: Apps for all, QtWayland reaches general availability
- **Mar 2015**: Mainline GNOME, GNOME 3.20 supports Wayland
- **Oct 2015**: wayland-protocols, Neutral extension development home
- **Sep 2015**: krh hands over, Pekka Paalanen releases 1.6

Open First
Wayland timeline: industrial revolution

- **Feb 2016**: Vulkan 1.0
  - Vulkan has day-zero Wayland support

- **Sep 2016**: Chrome Wayland
  - ARC++ used for Android in ChromeOS

- **Apr 2017**: Mir back to earth
  - Mir is retired, Ubuntu to use Wayland

- **May 2017**: xdg-shell stable
  - Window management protocol stable

- **Jun 2016**: libweston
  - Weston split into reusable library

- **Mar 2016**: sway 0.1
  - Will later be built into wlroots
Wayland timeline: late capitalism
History by numbers
Commits over time
Changes over time (LoC per month)
Commits over time (commit count as % of peak)
Where are we now?
Wayland in 2020

- Well, we basically won
- Multiple distributions shipping Wayland sessions by default
- Only realistic choice for single-use products (digital signage, IFE, IVI, kiosk, STB/DTV, etc)
- Mature offering but transition to maturity has been hard
- Rebuilding community around those lines
Where is the development?

- At first, fast and frantic development in central components
- Then it moved to the users as people implemented it
- Small period of re-centralisation as we came to mature the core and share extensions
- Most development now largely moved away back to users
- Core Wayland changes mostly limited to bugfixes
- Another period of heavy extension development probably coming
What did we do right?
Don’t be all things to all people

- Remember the goals and axioms of your project
- Keep your focus: concrete examples and use cases in mind
- Don’t be afraid to say no to scope creep!
- Early abstraction is death: compare X11/Wayland input systems
- But …
Don’t be all things to all people

- Leave yourself open to building out elsewhere
- Allow others to build on your common infrastructure and extend
- Wayland core is extremely light and doesn’t have so much functionality
- But lightweight stackable extension mechanism means other use cases can be supported outside the core
- If they’re a good idea, can bring them back in later
Be conscious of the world around you

- Optimise for your ideal world: for Wayland, this is the open-source ecosystem with upstream drivers
- Allow for others to experiment with your stack, maybe in a suboptimal way (e.g. fbdev, NVIDIA)
- Push out to other communities, make enablement happen
- Make it as easy as possible for users and developers to try it out
Extensibility and versioning

- Learn from us and Telepathy
- Have a good versioning strategy from the get go, which allows you to iterate and extend
- Make forwards compatibility easy, protocol breaks hard
- Define a clear deprecation path
Leadership transitioning

• Wayland was krh’s baby
• Stepping back was gradual and gentle, allowing other contributors to grow into larger roles
• If you weren’t looking closely, you almost wouldn’t notice he stepped away
• Project is still healthy and thriving
Burn down your own castle

- When we started, X11 was the only option: had all render, display, input, etc support tied up in it
- The history of Wayland is the history of generic Linux display infrastructure
  - GBM, KMS, dmabuf, Mesa EGL, libinput, libxkbcommon, libliftoff
- Anyone can now create their own window system
- Doing this keeps you honest: allows others to experiment outside your bounds, and challenge you
What did we do ... less right?
Clear and accessible information

- Reference and examples and narratives are all different
- Talk people through it visually
- Don’t have inaccessible and esoteric doc tools
- Presentations are really helpful to define your narrative
- But for concrete stuff, you need written docs
- Information vacuum is death: don’t let comment sections define your project
Toolkits beat midlayers

- libweston has the best compositor infrastructure
- But has historically had the worst user-facing API
- wlroots picked up enormous userbase from people who tried libweston and gave up
- Should’ve taken the lesson from the protocol: hand people toolkits they can use to build upon
Clear path for community

- Turns out if you have a really high barrier to entry, you get very few contributors
- Worked well in early days with focused corporate support from Collabora, Intel, Red Hat building Wayland from scratch
- Worked really badly with more diffuse community just trying to make Wayland work in their thing
- Transition to maturity was really hard
- Need to empower people where appropriate
Testing and validation plans

- Write tests, not test suites
- Build deep, not wide: make sure your test infrastructure is flexible and adaptable so building tests on top is easy
- Give people as few excuses to avoid testing as possible!
Know your time

• Wayland’s history covers many ages and stages of development and maturity
• Development approaches which were helpful in one age might not be helpful for the next
• Take a from-scratch look at your users and the world around you
• Make sure your processes are appropriate for your times
Thank you!