The Chromium/Wayland project

Maksim Sisov, msisov@igalia.com
Antonio Gomes, tonikitoo@igalia.com
Agenda

- Goals & Motivation
- Background
- Developments
- About the project
- Next steps
Goals
&
Motivation
Goal

- Be able to run Chromium natively on Wayland-based systems.
  - No XWayland,
  - No intermediate layers.
Motivation

- Wayland is a mature solution.
- Native Wayland support removes additional layer of abstraction compared to XWayland.
  - Less resource usage, less bugs.
- Demand from different industries.
  - Automotive, Mobile, Desktop
Background
Background - Ozone/Wayland

- Started by Intel / 01.org in 2014.
- Was based on initial Ozone project underneath Aura toolkit.
- Supported DRM/GBM for ChromeOS and Wayland for Linux (off trunk).
- Did not comply with Google’s vision on desktop integration.
Background - Ozone/Wayland

Desktop integration

Browser process
- desktop integration
- x11
- win

Renderer process

GPU process

Desktop integration (01.org)

Browser process
- desktop integration
- x11
- win

Renderer process

GPU process

ozone platform
wayland connection

IPC (old API)
Background - Ozone/Wayland

- Good community adoption.
- **Project entered in “maintenance mode”.**
  - December/2015.
  - Chromium m49.
    - Today’s ToT is **m67**.
In the meanwhile, Ozone layer in ToT received two new backends:
  ○ x11
  ○ wayland

The original “desktop integration” approach taken in Ozone/Wayland did not comply with the way future Linux desktop Chrome is foreseen.
Background - Cr Upstream (2/)

- **Ozone** project
  - Abstraction layer for the construction of accelerated surfaces **underlying the UI Service** (aka *Mus*), as well as input devices assignment and event handling.
  - Backends:
    - ChromeOS
      - DRM / GBM
      - x11
      - Wayland
    - Linux
Background - Desktop integration

Linux desktop integration (01.org)

- Browser process
  - desktop integration
  - x11
  - win
  - ozone/wayland
  - IPC (old API)

- GPU process
  - ozone platform
  - wayland connection

Mus Linux desktop integration

- Browser process
  - desktop integration
  - x11
  - win
  - aura/mus

- GPU service (gpu process)
  - Gpu service (thread)

- Window Server
  - ozone / wayland (connection)
  - ozone / x11

- UI Service
  - IPC (Mojo API)
New developments

Phase 1 - The bring up
Phase 1 - The bring up

- Sept-Oct/16
  - Igalia brought up of Ozone’s Wayland backend in ToT.
  - Experimented with “Ozone != ChromeOS”.
  - Documentation
  - Buildbots
Phase 1 - CrOS

- **Internal-window mode**
  - CrOS has a Window Manager (WM) and a ScreenManager (SM).
  - Chrome and other app windows in the system end up sharing a single display.
  - are embedded within a single top-level `acceleratedWidget`.
Phase 1 - Desktop Chrome

• External-window mode
  ○ Desktop Chrome has no WM.
    ■ One acceleratedWidget per Chrome window.
    ■ User manipulates acceleratedWidgets via the host OS window.
      • maximize, minimize, resizing, dragging, fullscreen.
  ○ Desktop Chrome has no SM.
New developments

Phase 2 - Chrome / Mus
Mus’ External Window Mode (1/)

- **Extended Mus and Ozone** to support ‘External Window’ mode:
  - Native `acceleratedWidget`’s for each top-level window.

- Ensured no major functionality loss if compared to stock Chrome.
Mus’ External Window Mode (2/)

- Added support to:
  - XDG v6.
  - Keyboard events, auto repeat, clipboard.
  - Mouse cursors.
  - Touch events.
  - Multiple windows.
  - Built-in window decoration.
  - Window closing.
  - Menus, widgets, and tooltips.
  - Support to common windowing features:
    - maximize, minimize, restore, fullscreen, dragging and resizing.
About the project (1/)

- The project is being hosted on GitHub.
- Well defined contribution policy:
  - Peer review.
  - Buildbot running existing tests ensuring no functionality loss.
About the project (2/)

- Rebase strategy:
  - Weekly based.
  - Continuous history clean up.
- Periodic sync up with Google.
- BlinkOn meeting (18th-19th of April):
  - Design discussion,
  - Upstream plans discussion.
About the project (3/)

- Documentation available at
  https://docs.google.com/document/d/1yzUWttsyqTh31vAyn4Xj4xblr3GOYIF44IBIFP_ixT0
Next steps

- Integration with AGL as web runtime.
- Continue upstreaming the project to ToT (clipboard, popup windows).
- General bug fixing.
- Decouple GPU service from the main browser process to gain additional 15% of performance gain (figures from Google engineers working on new VIZ service).
Questions?

msisov@igalia.com - Maksim Sisov

tonikitoo@igalia.com - Antonio Gomes