Wayland Shells for Embedded Systems

Michael Tretter – m.tretter@pengutronix.de

XDC 2023

Pengutronix.

About Me

- Michael Tretter
- Embedded Linux developer
- Pengutronix
- Graphics team

Embedded Systems

- Digital signage
- (Automotive) infotainment systems
- Industrial control panels
- Game consoles
- VR systems

Embedded vs. Desktop

- Resource constraints
- Custom input devices
- User interface defined by system developer

Common UI Layouts

- Single fullscreen application
- One application per output
- Split screen with two applications
- Picture in picture
- Fixed window layout
- Device-specific behavior

Window Management in Wayland

- Wayland compositor is the window manager
- System developer needs control over window management

Weston – Kiosk Shell

- Single fullscreen application per output
- Wayland clients use XDG shell protocol
- Identify clients by app ID

App IDs

- Defined in the XDG shell protocol
- The app ID identifies the general class of applications to which the surface belongs
- The compositor can use this to group multiple surfaces together, or to determine how to launch a new application

Weston – IVI-Shell

- In-Vehicle-Infotainment → Embedded
- Plugin to add custom behavior to Weston
- API for controlling the shell: ivi_layout_interface
- Wayland Client protocol: In-vehicle infotainment application
- Surprisingly large code base

AGL compositor

- Automotive Grade Linux → Embedded
- XDG shell protocol for applications
- AGL shell protocol for allowing clients certain control
- Client identification via app IDs
- Assumes a panel and background
- Built on libweston

Cage

- Single fullscreen application
- Application spans bounding box of all outputs
- No client identification necessary
- Built on wlroots

gamescope

- Single fullscreen X11 application
- Optimized for zero-copy (libliftoff)
- Color management
- No client identification necessary
- Built on wlroots
- Out of scope for the topic of this talk

phoc/phosh

- Wayland shell for phones
- Compositor and shell may run in processes
- Built on wlroots

qtcompositor

- Scripting and window placement with QML
- Wayland protocol callbacks are exposed to QML

Weston - Scriptable Window Management

- https://gitlab.freedesktop.org/wayland/weston/-/issues/520
- Weston desktop-shell plugin for Lua script
- Hooks call into script with top level surface

Custom Window Management?

- Are hooks for surfaces enough?
- Should there be some kind of API?
- Separate custom code from core compositor?

Client Identification?

- Are App IDs sufficient?
- What about multiple instances of the same program?
- GStreamer pipelines?

Wayland Client Protocol for Embedded?

- What do you miss in the XDG shell?
- Do graphic toolkits have to support the protocol?

Further Concerns

- Performance optimization and hardware planes
- Color management
- Screen recording and streaming
- Remote desktop protocol, VNC
- IPC, D-Bus, gRPC

Thank You!

Michael Tretter – m.tretter@pengutronix.de

Questions

- Custom Window Management?
- Client Identification?
- Wayland Client Protocol for Embedded?