AGL Compositor Architecture

Daniel Stone
daniels@collabora.com

COLLABORA

Hi, I'm Daniel

Graphics lead at Collabora
Open-source consultancy est. 2005
Wayland core developer

Open First

COLLABORA

Outline and agenda

Outline and agenda

- Share current AGL compositor architecture
- Window management API and concept
- OEM customisation
- Outline current progress and next plans

COLLABORA

Current compositor architecture

AGL compositor architecture

- Development has focused on window management and output management
- Outline window management concept and OEM API
- Outline homescreen development

Window management concept

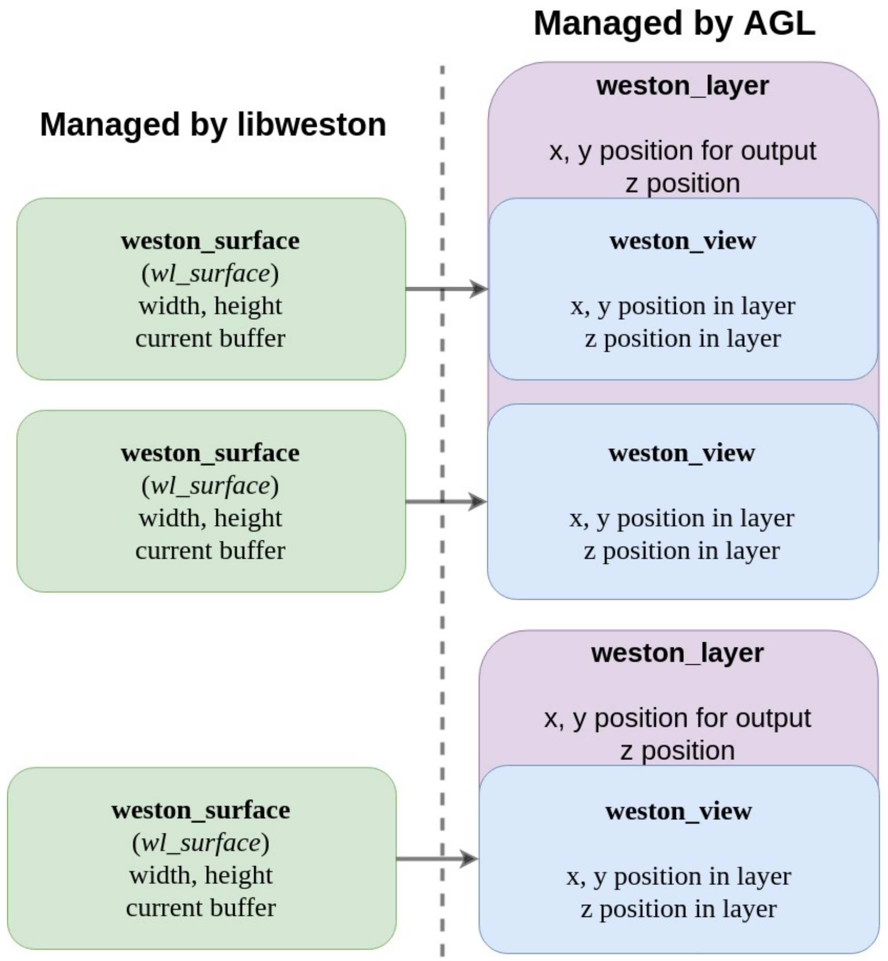
- WM based on output/layer/surface (like IVI shell)
- New concept from Weston: surface view
  - Views position an output within a layer
  - Multiple views allow to show surface in different places
  - Crucial for remoting: can create new view for other display or ECU
  - Window manager always controls views!

Window management concept

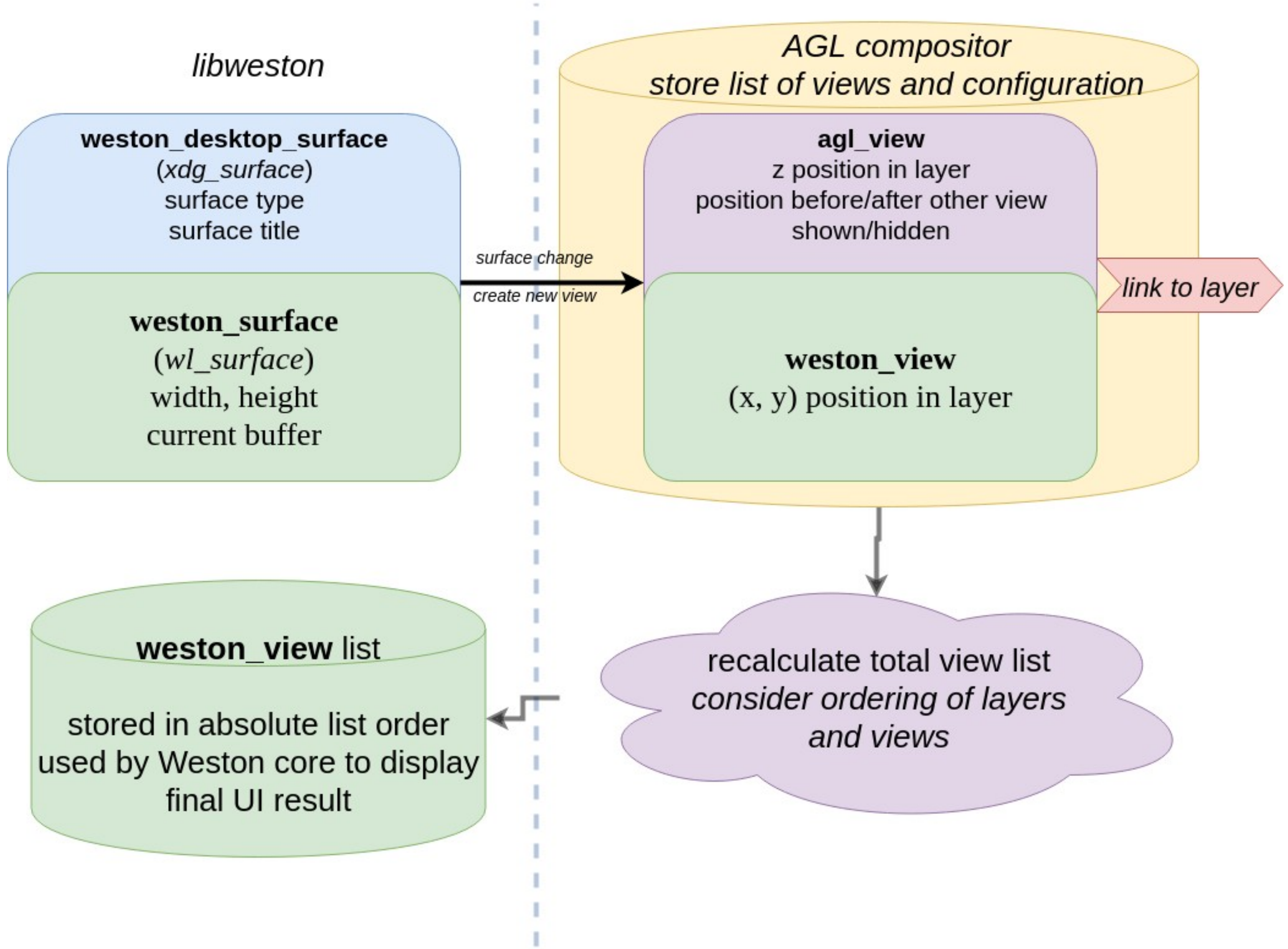
- Not so different from previous IVI shell!
- Key difference: give OEMs power to manage windows themselves with full API
- Offer callback into OEM module for every window event
  - new window created
  - window content updated
  - window removed

Surface/view relationship

- Compositor creates layers for grouping
- Positions layers within compositor space
- Compositor creates views for each surface to display
- Positions views within layers
- AGL IVI compositor API to manage view creation and positioning
- Display of views handled by libweston



Relationship between libweston and AGL views



Why two separate lists?

- Keep IVI concept of Z positioning
- Flexible positioning: allow views to be dynamically enabled/ disabled
- Easy integration with OEM WM policy
  - AGL view API can be stable for OEM plugins
- AGL core compositor will maintain translation between two worlds: recalculate libweston list after WM changes

Window management progress

- Core concepts implemented in working compositor
- Using IVI shell zpos concept
- AGL API to allow layers to be created, positioned, hidden
  - Layers can be dynamically added/removed
- AGL API to allow views to be created, positioned, hidden
  - Can be used by OEM WM policy plugins

Output configuration

- Basic output management compatible with Weston
- Allow outputs to be enabled/disabled, resolution set
  - depending only on output name
- More advanced output configuration API needed
- Weston already offers complex output configuration API
- Propose to have split APIs: simple and advanced
  - OEM can decide depending on usecase

COLLABORA

Compositor startup sequence

Compositor startup sequence

- The diagram doesn’t fit on a single slide ...
- Plan to reuse existing Weston documentation framework to include these diagrams with code documentation
- Produce HTML output for AGL documentation site
- ... and now to my browser

COLLABORA

Development plans

Window management & home screen

- Continue development of WM/HS implementation
  - Window management API largely in place
  - Home screen (AGL reference) porting WIP: end of July
  - Custom HS protocol to allow multiple windows
- Initial output management API implemented
- Next step after WM: app switching

Next work

- Aim to show functional home screen by end of July
- Should take ‘do not overwrite vendor logo’ requirement into account
- After home screen is complete, continue documenting WM/HS APIs for external users
- Develop input manager concept starting in August with support from others: hot keys, input routing
- Need separate topics in JIRA for all of these

Thankyou!

daniels@collabora.com