UI and Graphic EG

AGL would evolve as a Cockpit system more?
- To be collaborated with other Operation systems.
- To support more application.

- The reduction of lead time to integrate commercial application.
- Standard protocol Collaboration with OSs
  - Virtualization support

AGL Distribution releases for Single IVI system

AGL Distribution collaborates another ECU; E.g. Cluster Demo.
Standard protocol Collaboration with Various Operation systems.

An Operation system will be replaced by each OEM/Tiers for product
  • OS will be various
    • RTOS, Linux,
  • The way of physical will be various as well
    • MOST, Ethernet, LVDS, Virtualization shmem,

Goal: AGL distribution support standard protocol to be connected to Other OSs easily.

Solution:
  • Graphics:
    • Make the logical protocol as standard in Wayland/Weston
      (*) Wayland/Weston is now default of Fedora. Good place to contribute.
    • Contribute Plugin to hide physical.

<- DENSO as a UI and Graphics EG lead, would like to lead this activity.

(*) engineers who has an relationship with Wayland/Wesson needs
UI and Graphics Task

- Internode display protocol
  - Standard protocol Collaboration with Various Operation systems.
- Wayland update to 1.11
- Refactor Home Screen including splitting out window Manager
- Theming and skinning
- Improved PulseAudio and Audio Management configuration and policies.
- Replacement of QtMultimedia for media management?
- Speech services API and integration into reference apps.

This session

2/10 9:00-

2/8 session
By N.Nishiguchi
Graphics sharing protocol for multiple ECUs environment

Wataru Mizuno

Advanced Driver Information Technology Corporation

ADIT is joint venture company of DENSO Corporation and Robert Bosch GmbH/Robert Bosch Car Multimedia
Self introduction

- I worked for ICT Dept. DENSO Corp 2014-2016
  - I was in charge of development of future cockpit system
    - HMI to reduce driver’s distraction

- I started to work for ADIT since 2016
  - I am in charge of development of graphics middleware
    - Wayland / Weston
    - Wayland IVI extension
Information for drivers is increasing
- Increase of functions in each ECUs
  - Safety functions, telematics, V2X …
- Increase of displays
  - Increase of TFT Cluster
  - Increase of HUD (Head up display)

-> Too much information becomes distraction

- More comprehensive HMI is needed
  - Display information on appropriate place and time for driver
  - Comprehensive UI design

Graphics sharing between multi ECUs is needed
Realization method of sharing

Sharing between multi ECUs
  - Data are shared over network

- Sharing between guest OSs on virtualization environment
  - Data are shared on shared memory or over virtual network

Requirement: Same application is available on both single ECU and multi ECUs

Requirement: Same application is available on various kind of systems

Graphics sharing by using shared memory

Both graphic sharing mechanism and protocol are needed
Share navi map between IVI and Cluster

- IVI side navi map is shared to the center of the cluster side
We realized graphics sharing mechanism.
CES Cluster Demo future work

- Implement the protocol for sharing graphics between multi ECUs
  - IVI side send graphics information to Cluster side
    - Cluster side can change layout the kind of graphic
  - Cluster side send message complete rendering, request graphics…, to IVI side
    - Complete message is used for synchronization between multi ECUs
    - Request message is used when Cluster changes layout

- Share the graphics by surface
  - Good for sharing rear view camera, maps…
  - For standardization mechanism
    - Wayland treats graphics as surface

Not composited rear view camera surface
Composited rear view camera surface
Waltham : Wayland-style network IPC

- **Waltham enable surface sharing between multi ECU**
  - Waltham is IPC library of Wayland
  - https://github.com/waltham/waltham

- **Developed by Wayland community**
  - Many developers contribute this community and upstream patches
    - https://wayland.freedesktop.org/
  - Automotive: DENSO, BOSCH and other companies are using
  - DENSO and BOSCH are driving Waltham project with collaboration
Good points of Waltham

- **No special implementation to application needed for remoting**
  - Weston module “hmi-controller” decide share surface or not by surface ID
    - hmi-controller is weston standard module

- **Designed to adopt various structure of systems**
  - The physical layer can be suit for structure of system
    - Examples
      - Between remote ECUs: MOST, Ethernet AVB
      - Between guests OSs on VM: Shared memory

- **Waltham would be a standard protocol for graphics sharing**
Surface sharing architecture of Waltham

- **hmi-controller**: Handle HMI layout and remoting surface
- **Transmitter plugin**: Handle connection and transmit surface

ECU / Transmitter side

- IVI application
  - Weston core
  - Weston backend
  - displays
  - input devices

Decide remoting surface by surface ID.

ECU / Receiver side

- Transmitter plugin
  - It makes connection to Receiver side and transmit surface buffer, meta data and so on.
  - Receive buffer transmitted from Transmitter side.

Receiver

- Weston

Connection between ECUs is established by Waltham protocol.
Initialization Weston with Waltham

- The hmi-controller needs to start connection processing
- Receiver waits from transmitter connection
Start surface sharing

- The ivi-application does not need to do for remoting
  - The hmi-controller decide remoting surface by surface ID
Use case: IVI shares its surface with cluster on Multi ECU system

- IVI shares its surface of navi, cover art and so on, cluster gets IVI’s surface and composite it to cluster view

- Transmitter and Receiver can synchronize
- Receiver can request the kind of surface

---

### Diagram Description

**ECU**
- **IVI / Transmitter side**
  - **IVI application**
    - **Wayland**
    - **hmi-controller**
    - **ivi-shell**
    - **Transmitter plugin**
      - **Weston core**
        - **Displays**
        - **Input devices**

**ECU**
- **Cluster / Receiver side**
  - **Receiver**
    - **Wayland**
    - **Weston core**
    - **Displays**
    - **Ethernet**
      - **H.264**
    - **Control data and request data is sent**

**Surface encoded as H.264. Meta data and control data is sent**

Transmitter and Receiver can synchronize.
Receiver can request the kind of surface.
Use case: IVI shares its surface with cluster on Virtualization system

- IVI shares its surface of navi, cover art and so on, cluster gets IVI’s surface and composite it to cluster view

- Buffer address is shared to receiver side OS

**Diagram:**

- Guest OS1
  - IVI / Transmitter side
  - IVI application
  - Wayland
  - hmi-controller
  - ivi-shell
  - Weston core
  - Transmitter plugin

- Guest OS2
  - Cluster / Receiver side
  - Receiver
  - Virtual Network
  - Weston core

- Hypervisor
  - Shared memory
  - Buffer
  - Surface is shared via shared memory.
Conclusion

- Graphics sharing is needed for future cockpit system
- AGL cluster demo: Next step is implementation of protocol
- Waltham would be the best solution for graphics sharing
  - Standard protocol for sharing
  - Wayland application for single ECU is available as it is
  - Suit for various kind of systems

You can join Waltham project!
Unit of graphics on Wayland IVI Extension

**Surface**
Smallest parts of screen

**Layer**
Set of surfaces

**Screen**
Set of layers

© Advanced Driver Information Technology Corporation reserves all rights, including rights of disposal such as copying and passing on to third parties.
## Interface of Waltham

<table>
<thead>
<tr>
<th>Interface</th>
<th>Waltham original</th>
<th>Detail</th>
</tr>
</thead>
<tbody>
<tr>
<td>wth_display</td>
<td></td>
<td>In wayland protocol it is wl_display</td>
</tr>
<tr>
<td>wthp_registry</td>
<td></td>
<td>The Group of global objects. In wayland protocol it is wl_registry</td>
</tr>
<tr>
<td>wthp_callback</td>
<td></td>
<td>The callback interface. In wayland protocol it is wl_callback</td>
</tr>
<tr>
<td>wthp_compositor</td>
<td></td>
<td>The compositor. In wayland protocol it is wl_compositor</td>
</tr>
<tr>
<td>wthp_blob_factory</td>
<td>O</td>
<td>It creates wthp_buffer. Wthp_buffer is used for row pixel transfer</td>
</tr>
<tr>
<td>wthp_buffer</td>
<td></td>
<td>The buffer. In wayland protocol it is wl_buffer</td>
</tr>
<tr>
<td>wthp_surface</td>
<td></td>
<td>The surface. In wayland protocol it is wl_surface</td>
</tr>
<tr>
<td>wthp_seat</td>
<td></td>
<td>The group of input devices. In wayland protocol it is wl_seat</td>
</tr>
<tr>
<td>wthp_pointer</td>
<td></td>
<td>The pointer device. In wayland protocol it is wl_pointer</td>
</tr>
<tr>
<td>wthp_keyboard</td>
<td></td>
<td>The keyboard device. In wayland protocol it is wl_keyboard</td>
</tr>
<tr>
<td>wthp_touch</td>
<td></td>
<td>The touch screen device. In wayland protocol it is wl_touch</td>
</tr>
<tr>
<td>wthp_output</td>
<td></td>
<td>The Compositor output region. In wayland protocol it is wl_output</td>
</tr>
<tr>
<td>wthp_region</td>
<td></td>
<td>The region interface. In wayland protocol it is wl_region</td>
</tr>
</tbody>
</table>

- Receiver application should be implemented with the interface
- ivi-application does not use them
  - Transmitter absorbs the interface difference
- Easy to add interface by edit command.xml