Graphics Project & Distributed HMI

October 10, 2018 | GENIVI Technical Summit, Bangalore

Gunnar Andersson

Development Lead, GENIVI Alliance

Graphics Sharing and Distributed HMI

Project Goals

- All project participants gain thorough understanding of available choices
- Produce technology demonstrators, newly created or (if exists already) found and highlighted.
- Publish hard data on learning: Performance, resource needs.
- Seek industry acceptance & alignment among Linux distributions, as well as across operating systems and domains
- Seek alignment on solutions and protocols among proponents of “closed” alternatives – commercial HMI-tools, etc.
- Promote open standards and implementations across industry
- Separately identify and describe Hypervisor-based opportunities, how they differ, characteristics, advantages and disadvantages.
- Summarize and create (implementation) documentation for recommended choices

Definitions – Graphics Sharing

1) Graphics in the form of bitmap, scene graph, or drawing commands generated on one ECU and transferred for display by another ECU (or between virtual machine instances)

2) GPU sharing in a virtualized setup

Definitions – Distributed HMI Compositing

Methods and technologies to turn a multi-ECU system into what appears and acts as a single user-experience.

The 5 Categories of Graphics Sharing technologies

- GPU sharing
- Display sharing
- Surface sharing
  - Sub-category: **Virtual Display**.
- API Remoting
- Shared state, independent rendering

Reflections on GSHA as a domain-interaction example

Surface Sharing

- Operating systems exchange graphical (bitmap) content.

- Then, each OS has full flexibility to use this content.
- In some cases, the compositor API is made available remotely, e.g. Wayland->Waltham.

Surface Sharing example: Wayland, Waltham

- Wayland is a display server protocol for Linux
- Wayland defines how applications communicate their graphical content (surfaces) to a compositor that assembles (multiple) applications' combined appearance into the complete graphical screen output.
- On most computers this is a local communication between apps and system.
- Waltham enables Wayland to work over a network. (very simplified*) Since Wayland juggles "surfaces" - Waltham becomes an example of surface sharing

- **GSHA project wiki contains an analysis with much more depth*

Surface Sharing

- Other examples exist – probably numerous
- Proprietary HMI systems in particular
- Wayland is very Linuxy… can Waltham / Wayland protocol over network) become cross-platform standard?
  - GSHA topic: Study and compare to Android APIs
- Surface sharing with QNX? – See later case study

Virtual Display (surface sharing)

- The project considers this a sub-category to surface-sharing.

- Full display transfer by encoding as a video stream.

- Often characterized by a “transparent” API such that applications can use it as if it were a real display. (But the system still identifies “Virtual Display” as a separate object type – case in point Android API)

GPU Sharing

The GPU can be used from multiple operating systems, so it is shared. Concurrent access to the physical GPU has to be controlled by a hypervisor, hardware or other means which are implementation specific.

GPU Sharing

Considerations

- API standard?
- Unique hardware support?
  - Pass-through instead of virtualization?
- Portability across hardware standards
- Are standards feasible?
- → Working session tomorrow
  (Note: This is planned for the Hypervisor Working session)

Display Sharing

One physical display can be shared across multiple operating systems.

A HW compositor unit composites final display buffer from HW Layers of each OS.

This requires virtualization of the display controller hardware.

Hardware Layers

(contrast or complement: GENIVI Layer Management)

API Remoting

Transfer API calls, corresponding to "drawing commands", or other abstract graphics representation from one ECU to another. Commands or scene representation to be executed on the GPU of the receiving ECU.

“Remoting” existing APIs – (note the GPRO project for protocol evaluation)
or
Custom API for the task

API Remoting

Bandwidth efficient?

“It depends...”

Always, sometimes, in special cases only?

A current discussion point within the group

Shared State – Independent Rendering

Each system has independent graphics systems and bitmap information.
The systems only synchronize their internal state and exchange abstract data.

Based on this shared data, each system independently render graphics to make it appear like they are showing the same or related graphics.

Example: An appearance of synchronized map rendering could be achieved by drawing maps independently, and exchanging only the GPS position, scale of map, etc.

Shared State – Independent Rendering

Consequences...

Each participating system needs its own basic graphics (bitmaps, textures)

Data and rules for HMI look & feel must be aligned beforehand,

Software updates – required on both sides if look&feel changes

Navigation example: Both must have map data.

Advantages:

- The state/data transfer could be very small and bandwidth efficient

Shared State – Independent Rendering

Consequences...

Each participating system needs its own basic graphics (bitmaps, textures)

Data and rules for HMI look & feel must be aligned beforehand,

Software updates – required on both sides if look&feel changes

Navigation example: Both must have map data.

Advantages:
- The state/data transfer could be very small and bandwidth efficient

Thank you!

Visit GENIVI at http://www.genivi.org or http://projects.genivi.org

Contact us: help@genivi.org

First 2 Case Studies!

Case Studies (first session)

RAMSES : BMW Car IT (API Remoting) – **BMW**	**Violin Yanev (BMW)**
Android & Linux **Navigation interaction HMI** (Shared State) – **HARMAN**	**Sergey Klevitskiy (HARMAN)**

Case Studies (session two)

- **Qt studies : The Qt Company**
  - **Qt Remote Objects** (Shared State)
  - **Qt WebGL** (API Remoting)
  - **Qt WebAssembly** (API Remoting)
- **Implementing Waltham in practice : ADIT/Bosch** (Surface Sharing)
- **Android/QNX surface exchange : Harman** (Surface Sharing)
- **The Canvas-demo : Renesas** (Display Sharing, GPU sharing)
- **AllGo Multiple-display demo : AllGo** (Virtual Display)

GENIVI®