

# Answers for AGL Member's Comments for meta-uhmi

1. Any plans for maintenance of patches to Mesa, Weston, and the kernel.

- Mesa: Provides a workaround for the UHMI project by updating backing memory when buffer initialization.

- Weston: Add launcher-dummy to skip TTY setting in Unified HMI environment where weston is used over the virtio GPU over the network without VT.

- Kernel: Add card\_index into struct virtio\_device in kernel to set index id for virtual drm card device

➤ We plan to make adjustments so that it can be resolved with upstream versions.

# Answers for AGL Member's Comments for meta-uhmi

## 2. Comments for Recipes structure

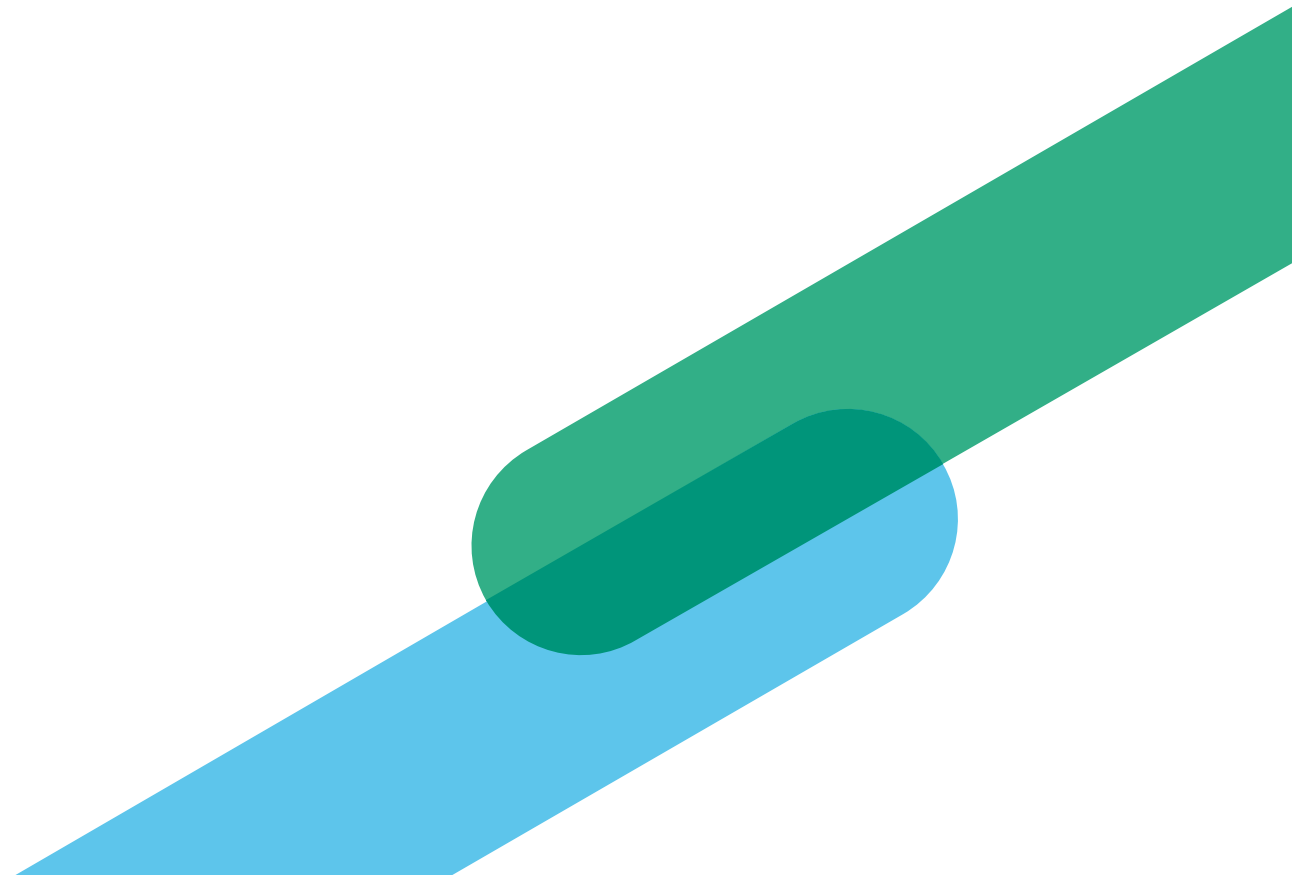
- The reason why meta-uhmi/meta-rvgpu.
  - This time, only RVGPU is committed, but eventually, we plan to commit Distributed Display Framework, and at that point, its recipe will also be placed under meta-uhmi.
  - Distributed Display Framework can be used individually when in single ECU system (no need RVGPU)
- About Panasonic-License-Agreement
  - We will remove Panasonic-License-Agreement file to follow to meta-agl-devel MIT license
- Place of agl-shell-desktop support patch for remote-virtio-gpu
  - We directly put this patch file in remote-virtio-gpu recipe instead of agldemo dynamic layers

# Answers for AGL Member's Comments for meta-uhmi

## 3. Comments for Recipes files

- Patch format regarding upstream status
  - We will add this flags to all patches.
- Remove the RPATH variable, to be able to use LD\_LIBRARY\_PATH
  - We would like to switch OpenGL library, so we need to set this configure use LD\_PRELOAD or some form of override or conditional include mechanism
- -Ddeprecated-wl-shell=true for weston
  - Now, rvgpu-renderer doesn't use xdg-shell, so we set this option to use wl-shell  
However, we will implement xdg-shell
- INSANE\_SKIP
  - We have already fixed local repository to remove INSANE\_SKIP.

**AGL All Member Meeting Unified HMI Presentation:  
Achieving a Software-Defined Multi-Display System  
with Unified HMI - Kenta Murakami, Panasonic  
Automotive Systems Co., Ltd.**

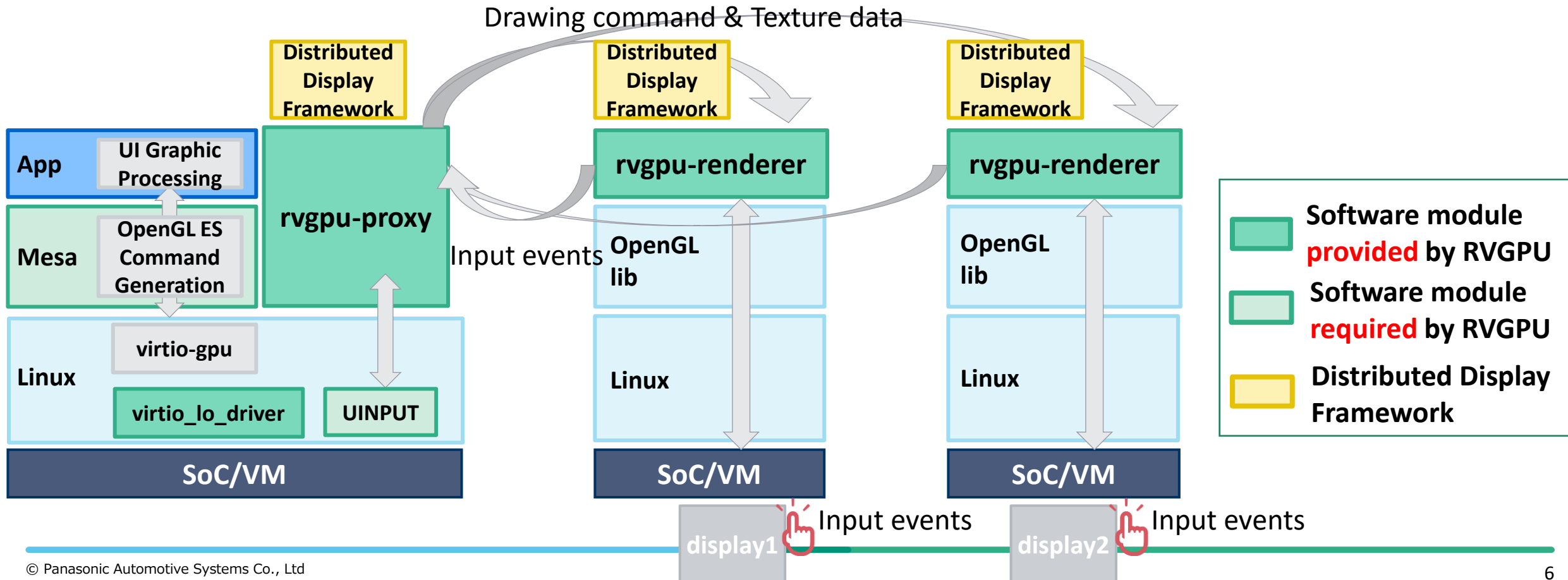


UNIFIED  OHMI

# Unified HMI architecture

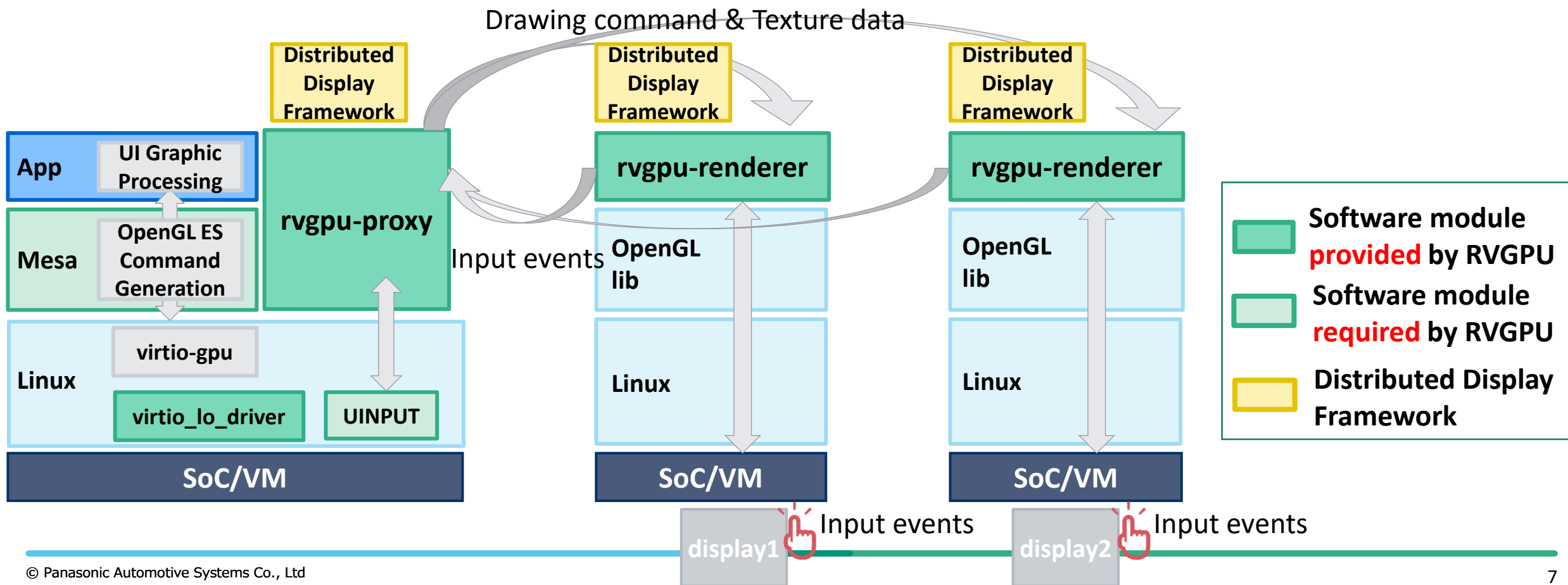
Consists of two main components.

1. **Remote Virtio GPU Device(RVGPU)** : Render apps remotely in different SoCs/VMs.
2. **Distributed Display Framework** : Flexible layout control of apps across multiple displays.



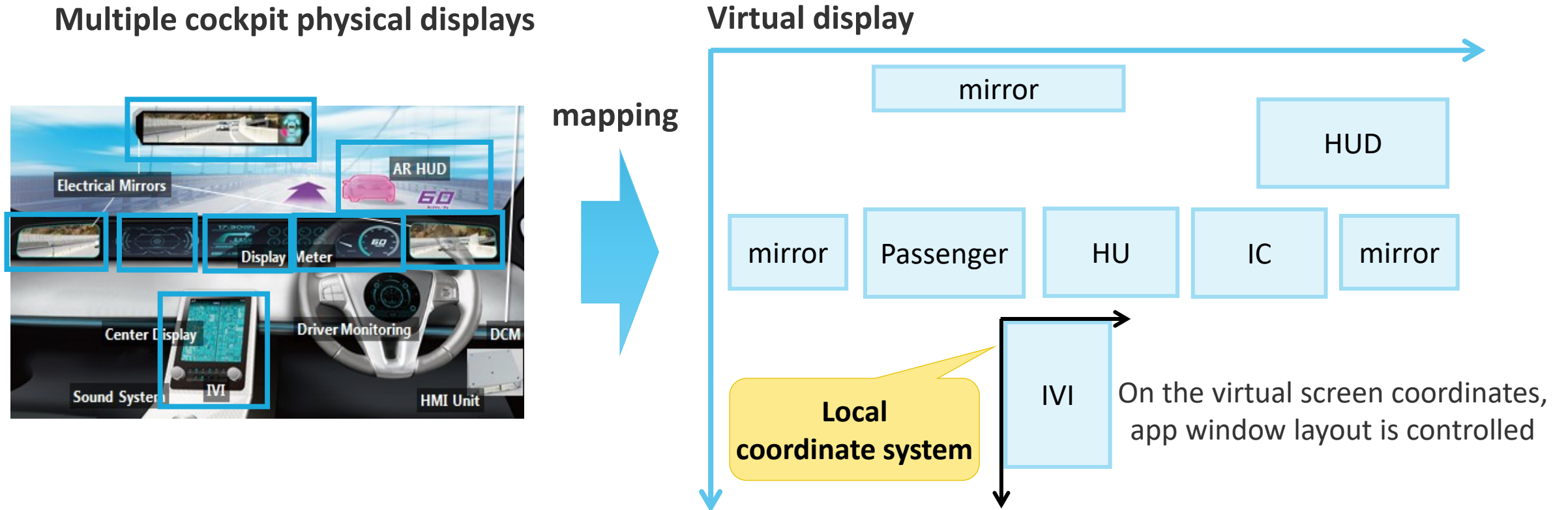
# 1. Remote Virtio GPU Device (RVGPU)

- > Network extension of **virtio-gpu** commonly used in GPU virtualization for VM.
- > **rvgpu-proxy** : Transfer GPU commands generated by OpenGL ES to other SoC/VM.
- > **rvgpu-renderer** : Receive GPU commands and draw graphics.







## 2. Distributed Display Framework

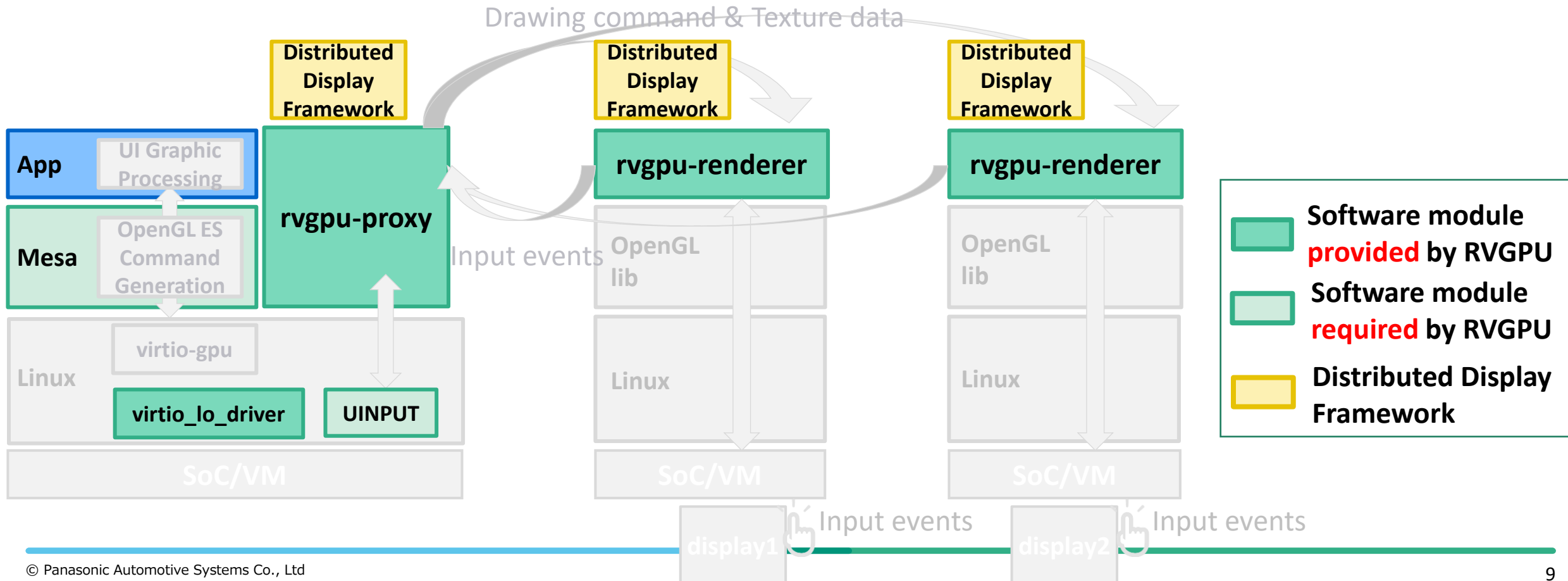
- Mapping multiple cockpit physical displays into a single large virtual screen.
- Control layout such as location, size, and display order of multiple apps.





# Unified HMI integration to AGL UCB

-   RVGPU (Already Committed)
-  Distributed Display Framework (By this year)
-  Flutter App support (By next year)



# Overview: How to use RVGPU on AGL

Step1. How to prepare your env.

Step2. How to download the AGL software

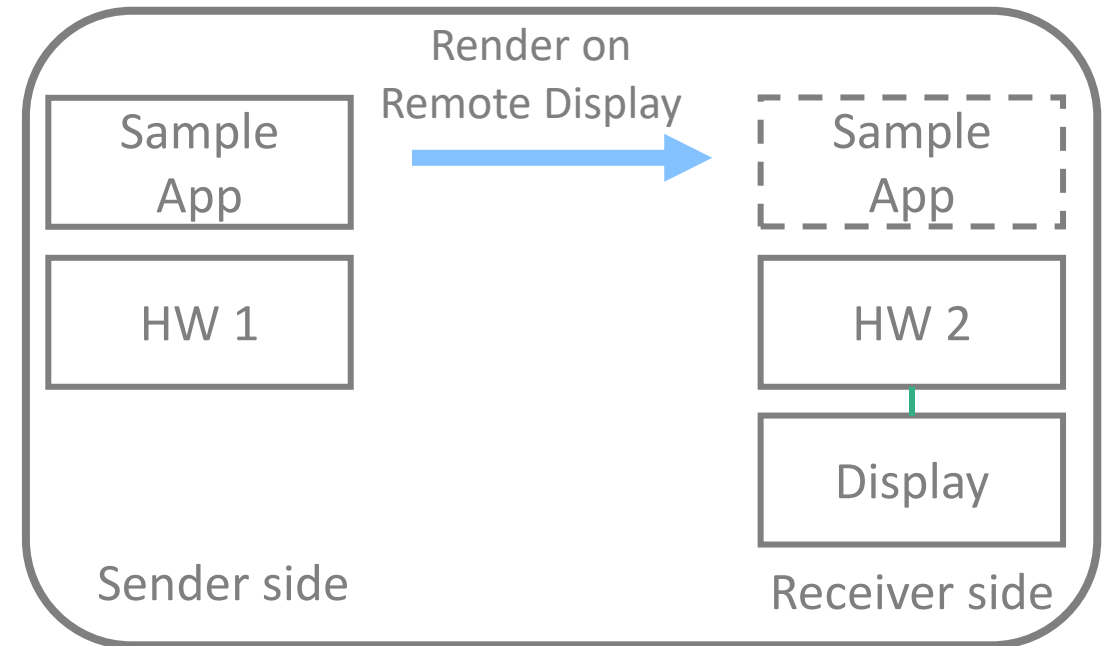
Step3. How to initialize your build env.

Step4. How to customize your build

Step5. How to build your image

Step6. How to deploy demo image

Step7. How to use RVGPU commands



# Overview: How to use RVGPU on AGL

Step1. How to prepare your env.

Step2. How to download the AGL software

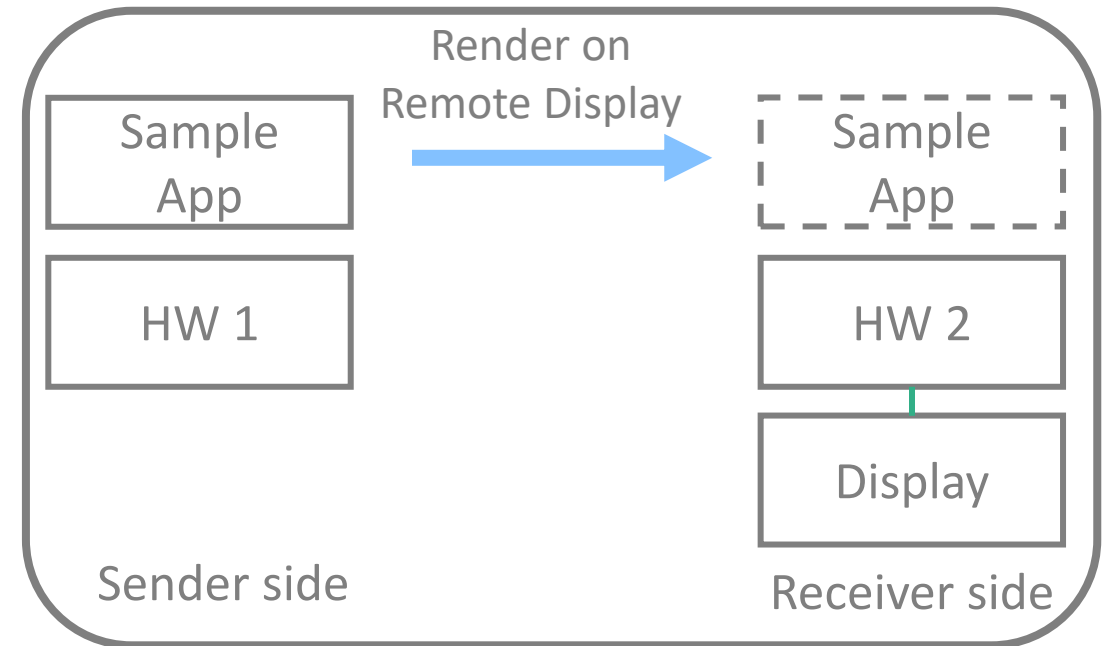
Step3. How to initialize your build env.

Step4. How to customize your build

Step5. How to build your image

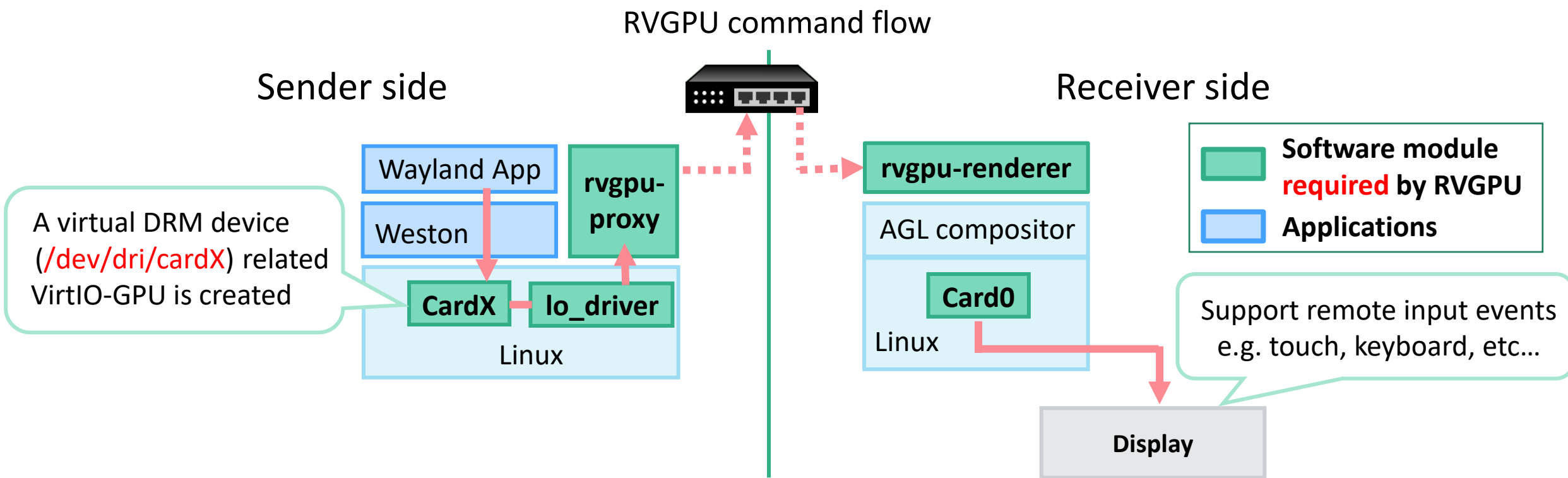
Step6. How to deploy demo image

Step7. How to use RVGPU commands



# Step1. How to prepare your env.

- > Support Platforms: x86(Emulation), Raspberry Pi4, AGL Reference Hardware
- > Prepare two or more of the above three platforms or Ubuntu PC and use one as Sender(where apps are running) and the others as Receiver(where app graphics are rendered).
- > All platforms used must be connected to the same network and accessible by unique IP Addresses.



# Overview: How to use RVGPU on AGL

Step1. How to prepare your env.

**Step2. How to download the AGL software**

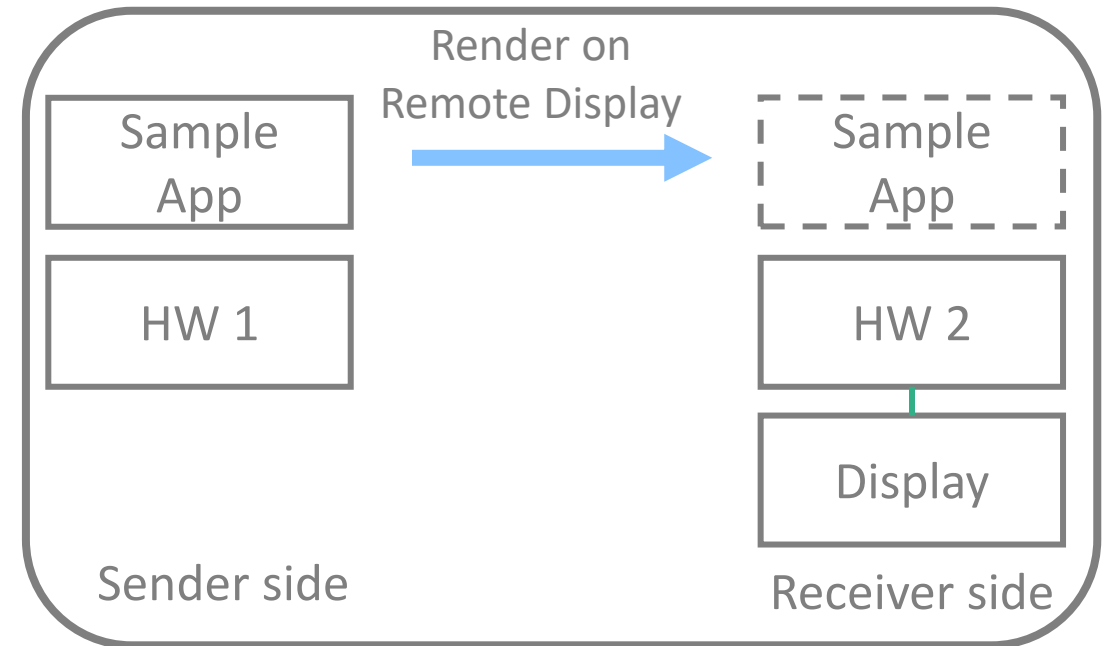
Step3. How to initialize your build env.

Step4. How to customize your build

Step5. How to build your image

Step6. How to deploy demo image

Step7. How to use RVGPU commands



## Step2. How to download the AGL software

- > Download the AGL software referring to AGL Documentation ([Downloading AGL Software - AGL Documentation \(automotivelinux.org\)](https://www.automotivelinux.org/))
- > Switch to the temporary branch under directory “meta-agl-devel”. (after community review and integration, correct path will be updated in Unified HMI related section in AGL Documentation)

```
$ cd meta-agl-devel
$ git fetch https://gerrit.automotivelinux.org/gerrit/AGL/meta-agl-devel
refs/changes/37/29037/1 && git checkout -b change-29037 FETCH_HEAD
```

- > After software is downloaded, the directory structure will be as follows.

```

${AGL_TOP}-/
├── meta-agl
├── ...
└── meta-agl-demo
    ├── meta-uhmi
    ├── ...
    └── meta-rvgpu

    ├── templates
    │   └── feature
    │       └── agl-rvgpu

```

# Overview: How to use RVGPU on AGL

Step1. How to prepare your env.

Step2. How to download the AGL software

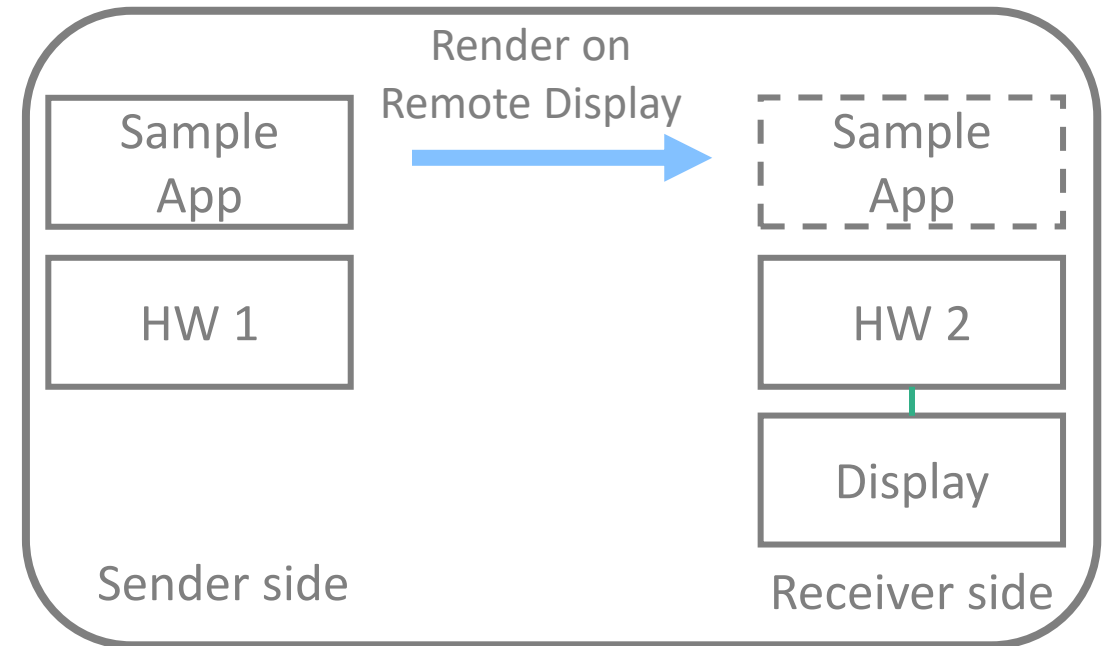
Step3. **How to initialize your build env.**

Step4. **How to customize your build**

Step5. **How to build your image**

Step6. **How to deploy demo image**

Step7. How to use RVGPU commands







# Overview: How to use RVGPU on AGL

Step1. How to prepare your env.

Step2. How to download the AGL software

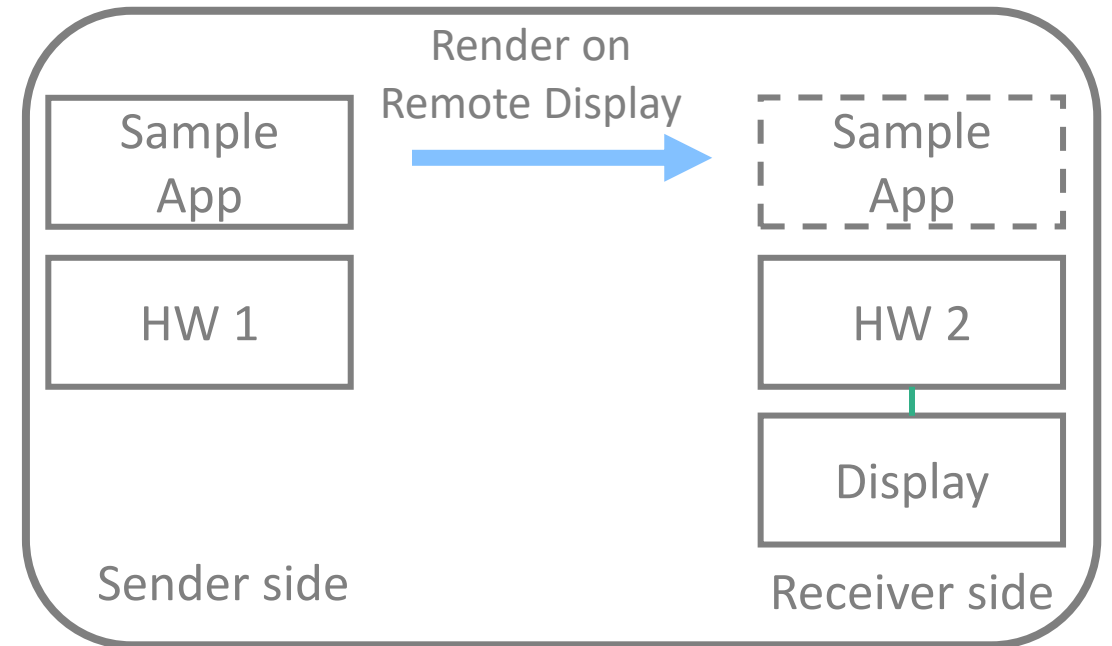
Step3. How to initialize your build env.

Step4. How to customize your build

Step5. How to build your image

Step6. How to deploy demo image

**Step7. How to use RVGPU commands**





# Future vision

## ① Extension to support Flutter Apps

*Supported App type*

Qt



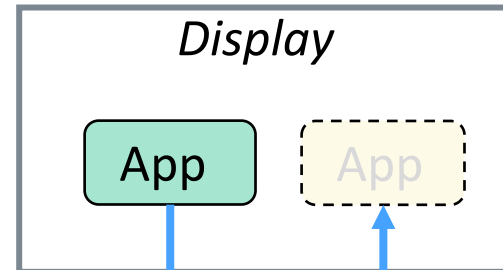
[qt icon by icons8](#)

Flutter



[flutter icon by icons8](#)

## ② Enable Application Graphics to Render In/Out between AGL and other OSES



Render Out  
App Graphic

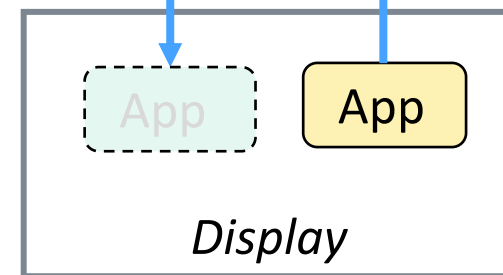
Render In  
App Graphic



Linux



Android



## ③ Extend Unified HMI to more media (audio, video and etc.)

*Supported Device type*

Graphic



Audio



Video



etc...