

IVI Graphic Subsystem: Weston IVI-shell is ready for Product

Nobuhiko Tanibata

1st July 2014

■ Introduction

- ◆ Trends of Graphic stacks in IVI segment
- ◆ Problems & Solution

■ Details of Wayland-ivi-extension and ivi-shell

■ Protocol: ivi-application and ivi-controller

■ Current Status

- ◆ TIZEN IVI
- ◆ GENIVI
- ◆ SoCs

■ Next Steps till Next AGL/GENIVI All member meeting

- ◆ Security model
- ◆ Adaptation of common Application Framework
- ◆ etc

■ Demonstration

- **Proprietary to Common**
- **Complexity to Light weight**
- **Wayland/ Weston is one of candidates**
 - ◆ Distill out functions from X server



- **Trends**
 - ◆ Tizen IVI: 2014 M1
 - ◆ GENIVI: 2014 GENIVI AMM
 - ◆ many companies shift proprietary stacks to using Wayland/ Weston

Problem & Solution : Wayland/Weston to IVI

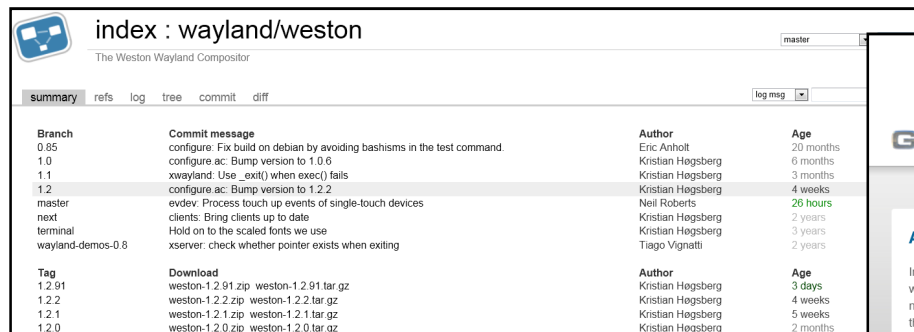
Advanced Driver
Information Technology

■ Problem:

- ◆ Coverage of requirements for IVI/Automotive system
- ◆ Quality level on IVI/Automotive system

■ Solution: IVI shell on Weston and GENIVI Layer manager APIs

- ◆ Support of GENIVI Layer Manager APIs
 - a set of APIs to cover requirements for IVI/Automotive system.
- ◆ Contribute it to major IVI distribution to be used by many users
 - TIZEN IVI
- ◆ Integrate it to Actual product
 - Qualified in product use uses



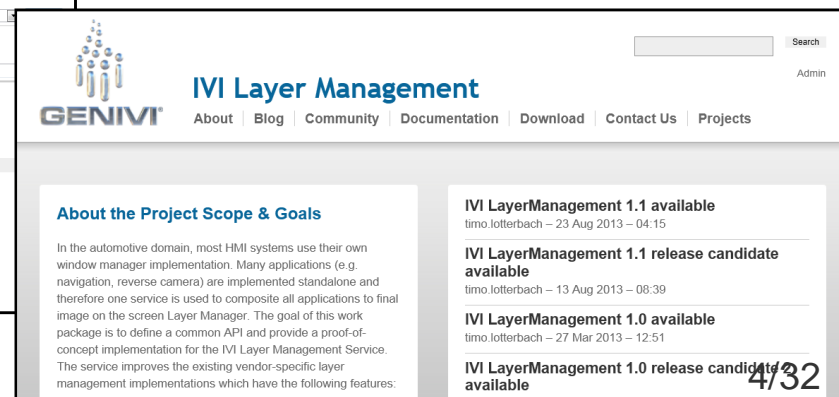
index : wayland/weston

The Weston Wayland Compositor

summary refs log tree commit diff

Branch	Commit message	Author	Age
0.85	configure: Fix build on debian by avoiding bashisms in the test command.	Eric Anholt	20 months
1.0	configure.ac: Bump version to 1.0.6	Kristian Hogsberg	6 months
1.1	xwayland: Use _exit() when exec() fails	Kristian Hogsberg	3 months
1.2	configure.ac: Bump version to 1.2.2	Kristian Hogsberg	4 weeks
master	evdev: Process touch up events of single-touch devices	Neil Roberts	26 hours
next	clients: Bring clients up to date	Kristian Hogsberg	2 years
terminal	Hold on to the scaled fonts we use	Kristian Hogsberg	3 years
wayland-demos-0.8	xserver: check whether pointer exists when exiting	Tiago Vignatti	2 years

Tag	Download	Author	Age
1.2.01	weston-1.2.01.zip weston-1.2.01.tar.gz	Kristian Hogsberg	3 days
1.2.2	weston-1.2.2.zip weston-1.2.2.tar.gz	Kristian Hogsberg	4 weeks
1.2.1	weston-1.2.1.zip weston-1.2.1.tar.gz	Kristian Hogsberg	5 weeks
1.2.0	weston-1.2.0.zip weston-1.2.0.tar.gz	Kristian Hogsberg	2 months



GENIVI IVI Layer Management

About | Blog | Community | Documentation | Download | Contact Us | Projects

Admin

About the Project Scope & Goals

In the automotive domain, most HMI systems use their own window manager implementation. Many applications (e.g. navigation, reverse camera) are implemented standalone and therefore one service is used to composite all applications to final image on the screen Layer Manager. The goal of this work package is to define a common API and provide a proof-of-concept implementation for the IVI Layer Management Service. The service improves the existing vendor-specific layer management implementations which have the following features:

IVI LayerManagement 1.1 available

timo.lotterbach – 23 Aug 2013 – 04:15

IVI LayerManagement 1.1 release candidate available

timo.lotterbach – 13 Aug 2013 – 08:39

IVI LayerManagement 1.0 available

timo.lotterbach – 27 Mar 2013 – 12:51

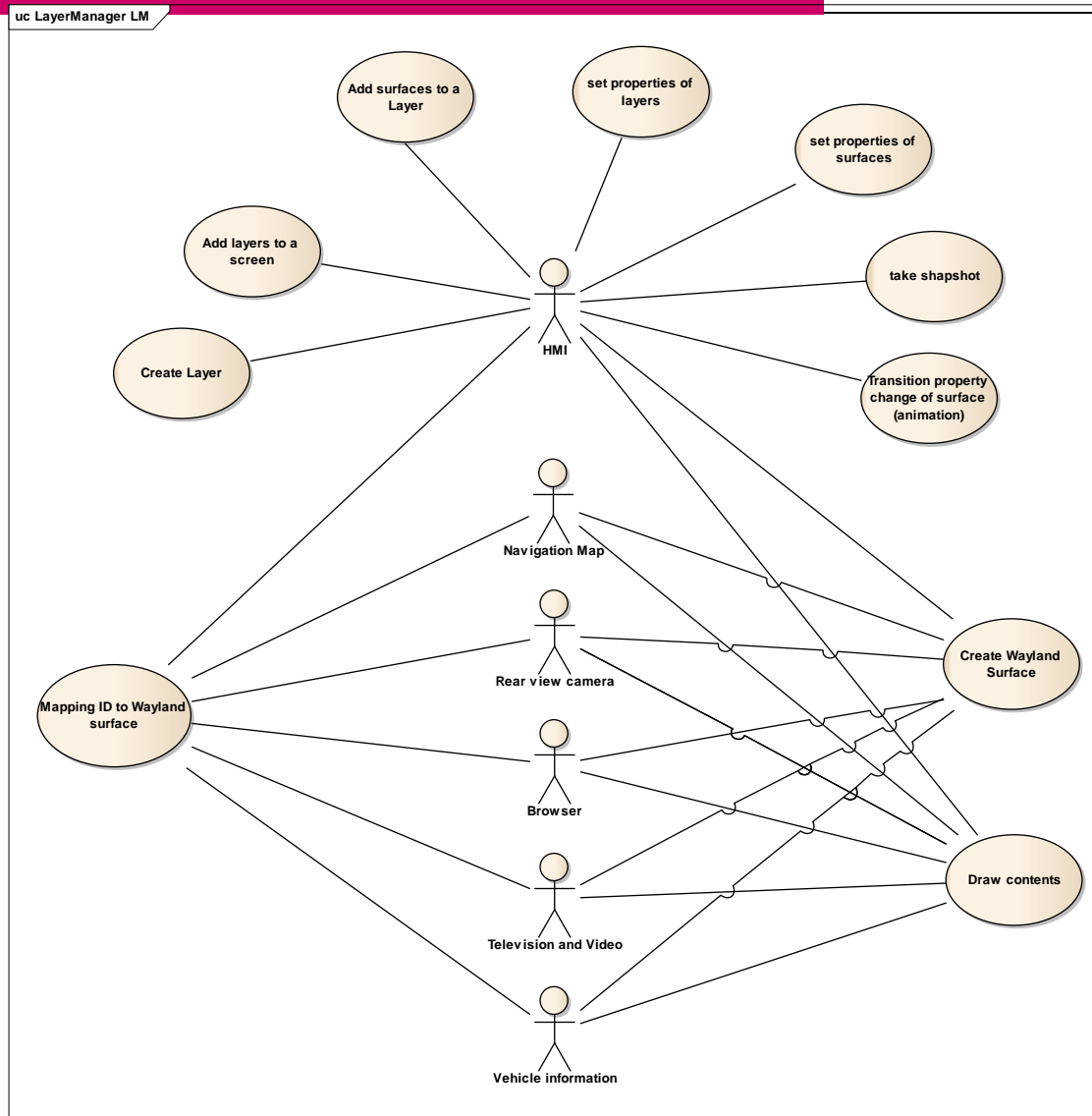
IVI LayerManagement 1.0 release candidate available

timo.lotterbach – 10 Mar 2013 – 10:40

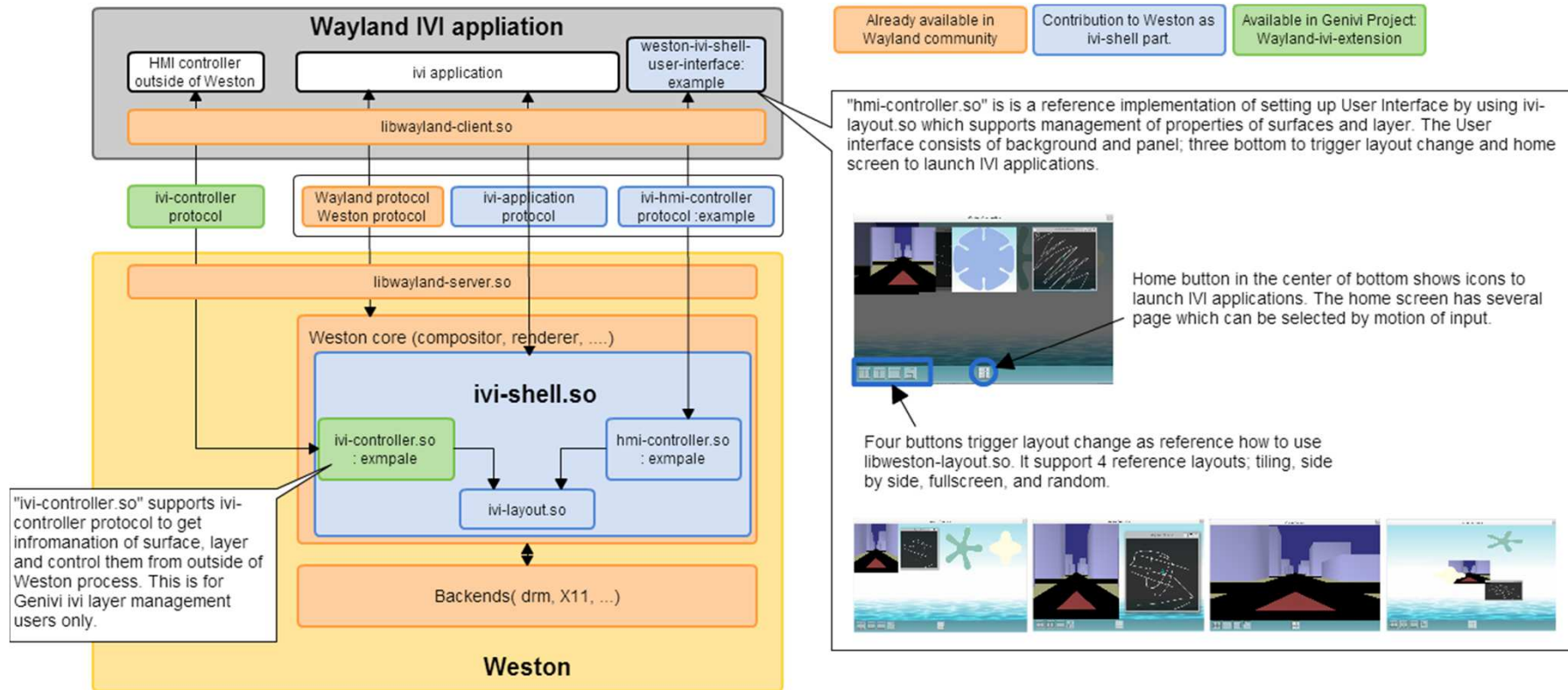
Weston IVI shell and Layer manager APIs

Use cases for IVI shell and Layer manager APIs

Advanced Driver Information Technology

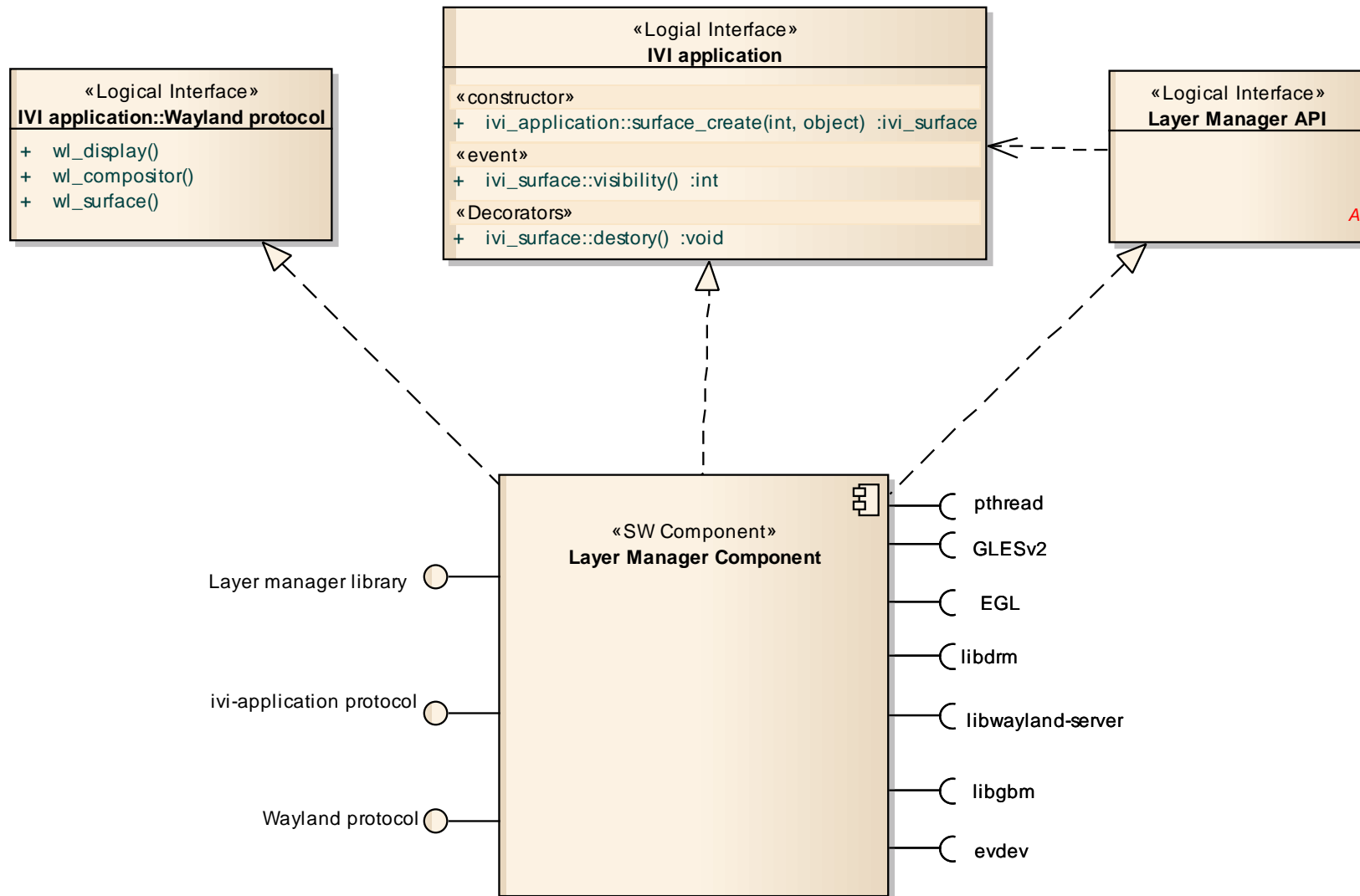


Overview of IVI shell and Layer manager APIs

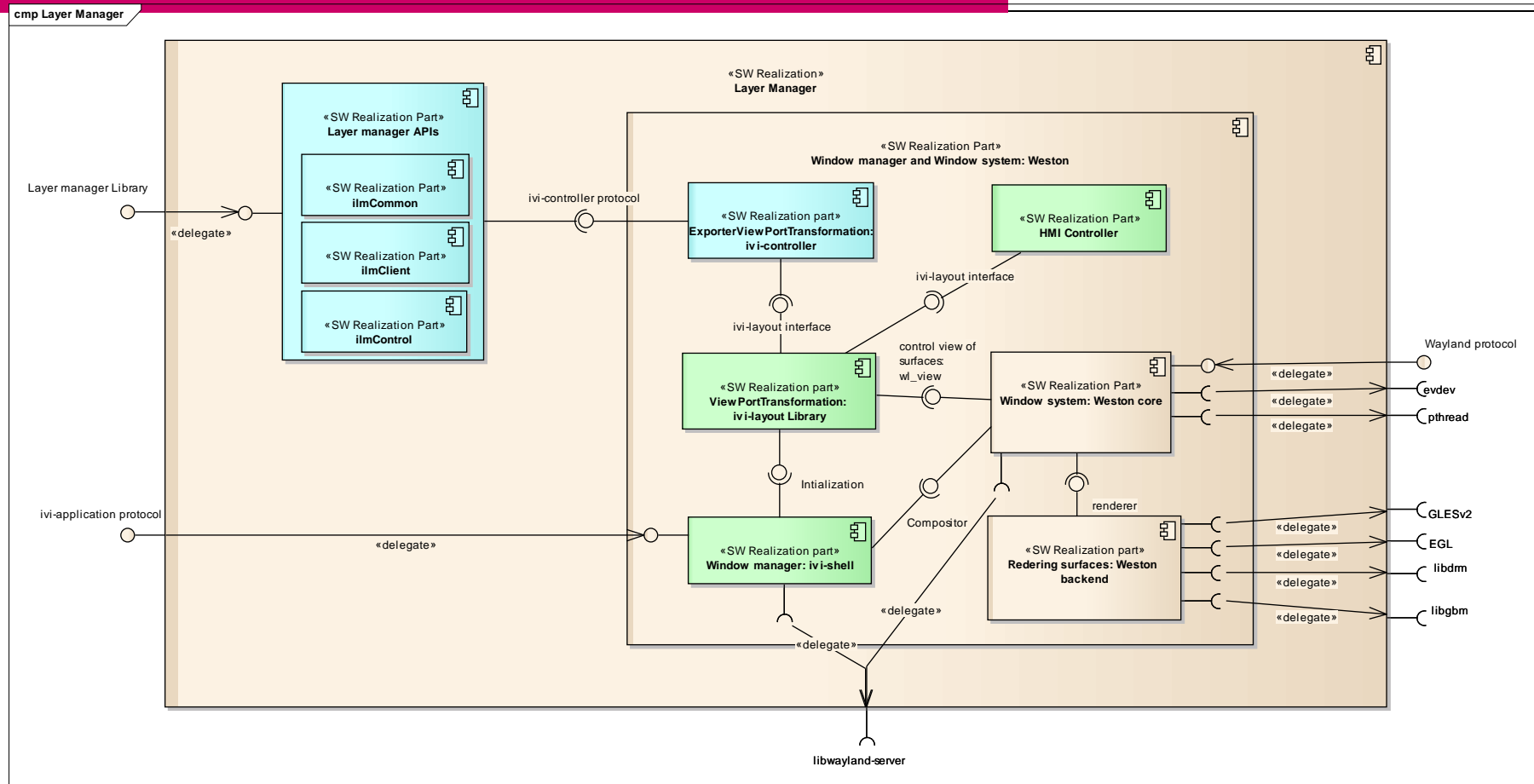


Interfaces

cmp Layer Manager

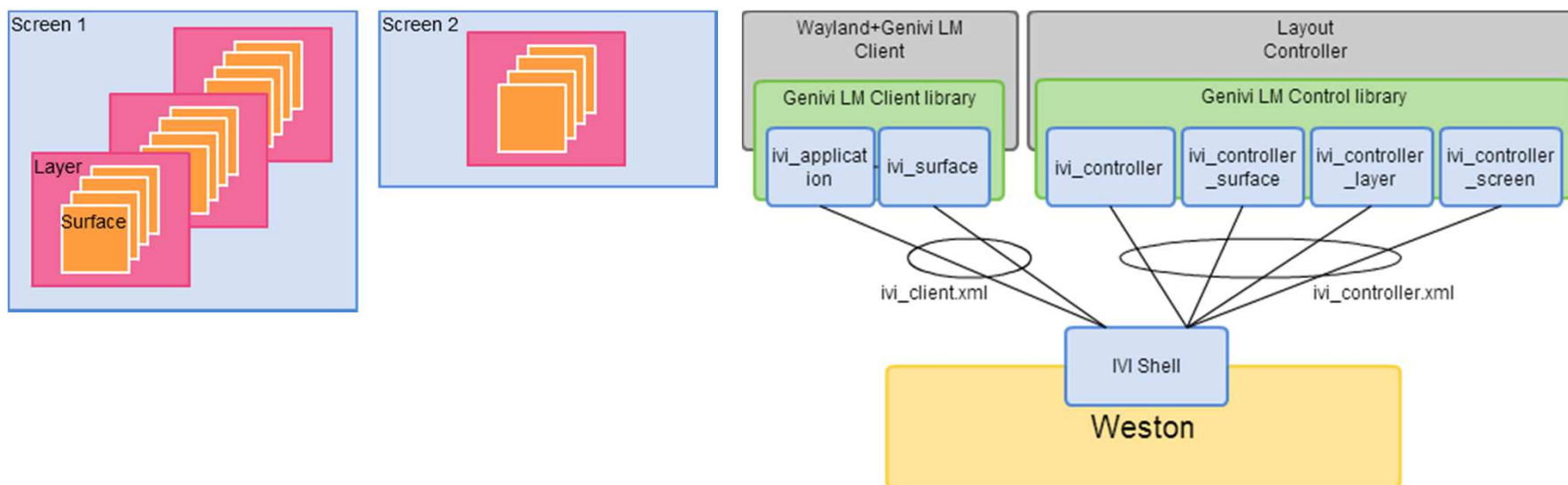


Component diagram



Protocol: ivi-application And Ivi-controller

- Define IVI specific protocol to fit GENIVI layer management; managing surface->Layer->Screen.
- Clearly define a role of application and controller.

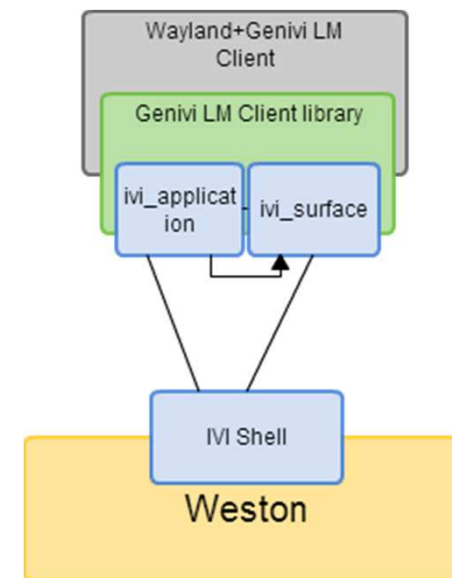
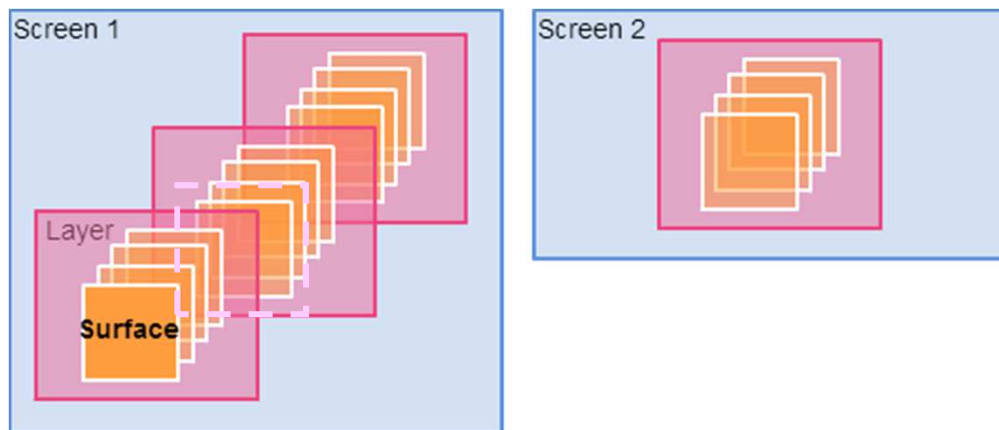


Use case: Wayland application set its native to ivi_surface

- **ivi_application**: the first protocols for creation of surface.
- **ivi_surface**: set weston native_handle to ivi_surface

Simple protocol to tie native and ivi_surface with global ID.

Global ID allow us to identify ivi_surface.

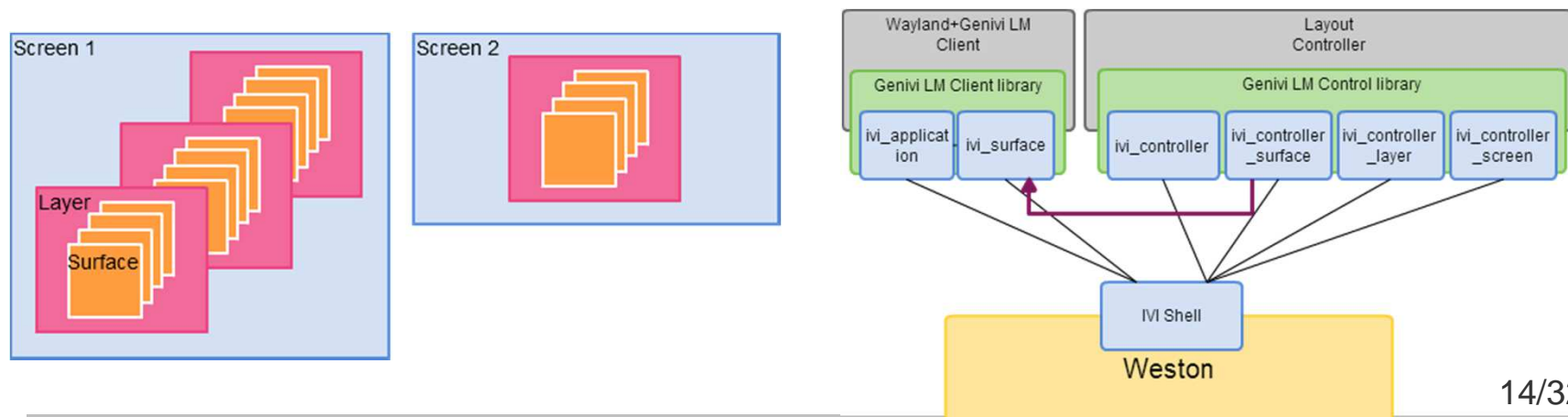


- **<interface name="ivi_application" version="1">**
 - ◆ **<request name="surface_create">**
 - **<arg name="id_surface" type="uint"/>**
 - **<arg name="surface" type="object" interface="wl_surface"/>**
 - **<arg name="id" type="new_id" interface="ivi_surface"/>**
 - ◆ **</request>**
- **</interface>**

Simple interface is supported to map wl_surface to Global ID.

Use case: Create layers, add surfaces to it and control them.

- **ivi_controller**: the first protocols for receiving events: creation of surface and create layer.
- **ivi_controller_surface**: set visibility e.g. in case of speed restriction.
- **ivi_controller_layer** : add/clear surfaces, set visibility, position,
- **ivi_controller_screen**: add layer to a screen



ivi-controller.xml: ivi_controller

```
<protocol name="ivi_controller">

  <interface name="ivi_controller" version="1">
    <description summary="Interface for central controller of layers and surfaces"/>
    ...
    <request name="layer_create">
      <description summary="ilm_layerCreateWithDimension"/>
      <arg name="id_layer" type="uint"/>
      <arg name="width" type="int"/>
      <arg name="height" type="int"/>
      <arg name="id" type="new_id" interface="ivi_layer"/>
    </request>

    <event name="layer">
      <description summary="Receive id_layer/ivi_layer and a controller to control ivi_layer"/>
      <arg name="id_layer" type="uint"/>
      <arg name="layer" type="new_id" interface="ivi_layer"/>
      <arg name="controller" type="new_id" interface="ivi_controller_layer"/>
    </event>

    <event name="surface">
      <description summary="Receive id_surface/ivi_surface and a controller to control ivi_surface"/>
      <arg name="id_surface" type="uint"/>
      <arg name="surface" type="new_id" interface="ivi_surface"/>
      <arg name="controller_surface" type="new_id" interface="ivi_controller_surface"/>
    </event>
    ...
  </interface>
</protocol>
```

ivi-controller.xml: ivi_controller_surface

```
<protocol name="ivi_controller">

  <interface name="ivi_controller_surface" version="1">
    <description summary="Request property change of ivi_surface to server"/>
    ...

    <request name="set_visibility">
      <description summary="Set Visibility"/>
      <arg name="visibility" type="uint"/>
    </request>

    <event name="visibility">
      <description summary="sent in response to set visibility"/>
      <arg name="visibility" type="int"/>
    </event>

    ...

    <event name="layer">
      <description summary="Receive a ivi_layer this ivi_surface belongs"/>
      <arg name="layer" type="object" interface="ivi_layer" allow-null="true"/>
    </event>

    ...
```



```
<protocol name="ivi_controller">
```

```
  <interface name="ivi_controller_layer" version="1">
```

```
    <description summary="Request property change of ivi_layer and add/remove ivi_surface from ivi_layer to server"/>
```

```
    ...
```

```
    <request name="set_visibility">
```

```
      <description summary="Set Visibility"/>
```

```
      <arg name="visibility" type="uint"/>
```

```
    </request>
```

```
    ...
```

```
    <request name="add_surface">
```

```
      <description summary="add a ivi_surface to top order of a ivi_layer"/>
```

```
      <arg name="surface" type="object" interface="ivi_surface"/>
```

```
    </request>
```

```
    ...
```

```
    <event name="screen">
```

```
      <description summary="Receive a wl_output this ivi_layer belongs"/>
```

```
      <arg name="screen" type="object" interface="wl_output" allow-null="true"/>
```

```
    </event>
```

```
    ...
```

ivi-controller.xml: ivi_controller_screen

```
<protocol name="ivi_controller">
```

```
  <interface name="ivi_controller_screen" version="1">
```

```
    <description summary="Request add/remove layer from ivi_layer to server"/>
```

```
    ...
```

```
    <request name="add_layer">
```

```
      <description summary="add a ivi_layer to top order of a wi_output"/>
```

```
      <arg name="layer" type="object" interface="ivi_layer"/>
```

```
    </request>
```

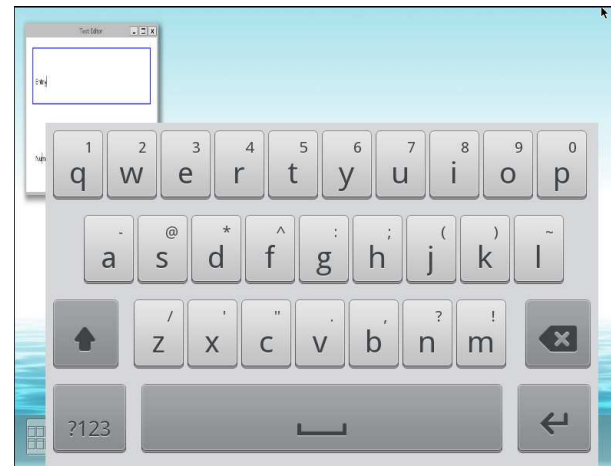
```
  ...
```

Current Status

■ GENIVI Layer Manager APIs

■ New from weston-ivi-shell

- ◆ Wayland protocol except other shell type; `wl_shell` and `wl_get_shell`
- ◆ Input method
- ◆ Transition animation



Where is code?

Project page in GENIVI:

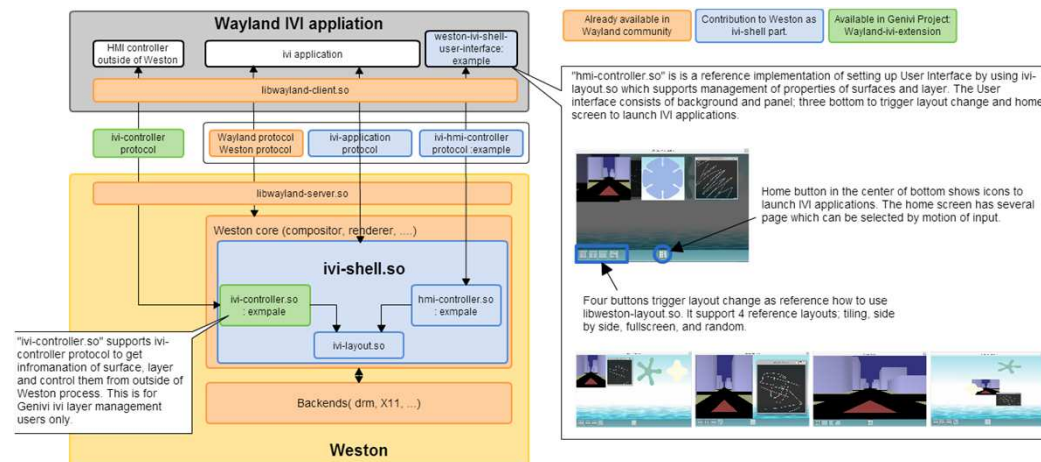
<http://projects.genivi.org/wayland-ivi-extension/>

■ Weston ivi-shell; weston 1.5 patches

- ◆ <https://github.com/ntanibata/weston-ivi-shell/tree/weston-ivi-shell-1.4.93-v3>

■ Wayland ivi extension

- ◆ <http://git.projects.genivi.org/?p=wayland-ivi-extension.git;a=summary>



URL: <http://projects.genivi.org/wayland-ivi-extension/documentation>

■ Weston with ivi shell

- ◆ Pre-requisition: wayland and other component.
- ◆ git clone source and autogen.sh. If there is missing dependency, “configure” will point it.

■ Wayland ivi extension

- ◆ Pre-requisition: weston with ivi-shell
- ◆ git clone source and cmake ./ -DBUILD_ILM_API_TESTS=1

■ A reference Weston.ini: <build dir>/ivi-shell/weston.ini

◆ Copy it in \$HOME/.config/weston.ini

- [core]
- shell=ivi-shell.so

- [ivi-shell]
- ivi-module=hmi-controller.so
-

■ Execute “weston”

- ◆ export XDG_RUNTIME_DIR=/var/run/user/1000
- ◆ /usr/bin/weston

- **ivi-shell upstream: Weston 1.5.1 (plan)**
 - ◆ Except. To be patched later.
 - Transition animation
 - input method
- **GENIVI: Wayland-ivi-extension:**
 - ◆ sanity test available

Thanks to Kritian, Intel and Layer manager Team!!

■ One of collaboration project

- ◆ https://wiki.tizen.org/wiki/IVI/Tizen-IVI_3.0-M2-March2014
- ◆ Wayland-ivi-extension and ivi-shell are integrated
 - <https://review.tizen.org/git/?p=profile/ivi/weston-ivi-shell.git;a=summary>
 - <https://review.tizen.org/git/?p=profile/ivi/wayland-ivi-extension.git;a=summary>
- ◆ Sample HomeScreen (ICO) with layer management
 - <https://review.tizen.org/git/?p=profile/ivi/ico-uxf-homescreen.git;a=summary>
 - Developed by TOYOTA daughter company, TTDC

Thanks to TIZEN IVI team. Especially strong support from Ossama Othaman, Intel.

- **No dependent on Architecture and run on major SoCs**
 - ◆ Intel: Baytral
 - ◆ nVIDIA: TegraK1
 - ◆ Renesas: R-Car M2
 - ◆ Freescale: imx6

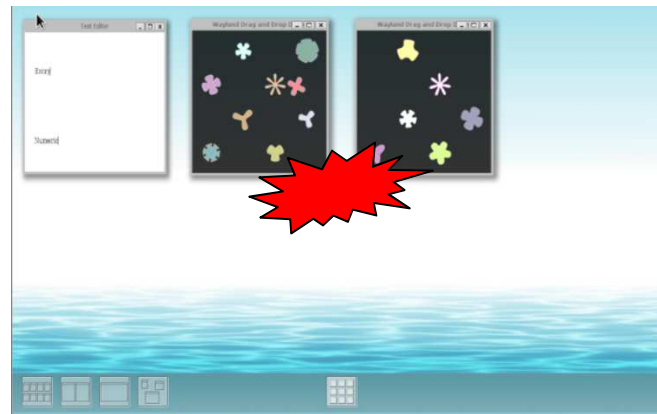
Next Steps

■ ivi-controller protocol shall not be bound by all applications.

- ◆ Mal-application will damage a Scene graph.
- ◆ The worst case e.g. Television application can show the surface during speed restriction.

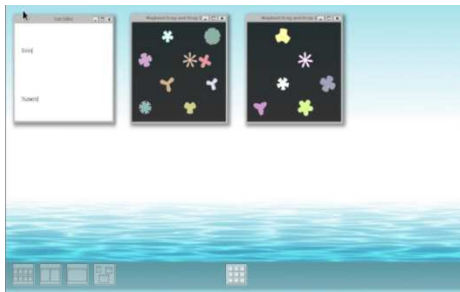
■ Proposal

- ◆ Only processes forked by ivi-controller are allowed to control surface/layer.
- ◆ Or white list or black list



- If a application framework already support wl_shell or xdg_shell, it might be easy.
- Which is preferable?
 - ◆ Qt
 - ◆ HTML5
 - ◆ Any feedback?

- **An application likes to request the shape of its surface**
 - ◆ Full screen
 - ◆ Half size of screen. E.g. show route guidance.
 - ◆ Popup
 - ◆ Etc
 - ◆ However, it would be just request. Final decision shall be done by ivi-controller.
- **Proposal**
 - ◆ New request needs to be define. Like set_behavior.



- **An application want to receive events e.g. steering switches even if it is located on the top.**
- **Proposal**
 - ◆ GENIVI Layer management APIs defines input focus API
 - ◆ Focus surface per input devices. E.g. a surface for multi media is set to be focused to receiving volume up/down from steering switches.

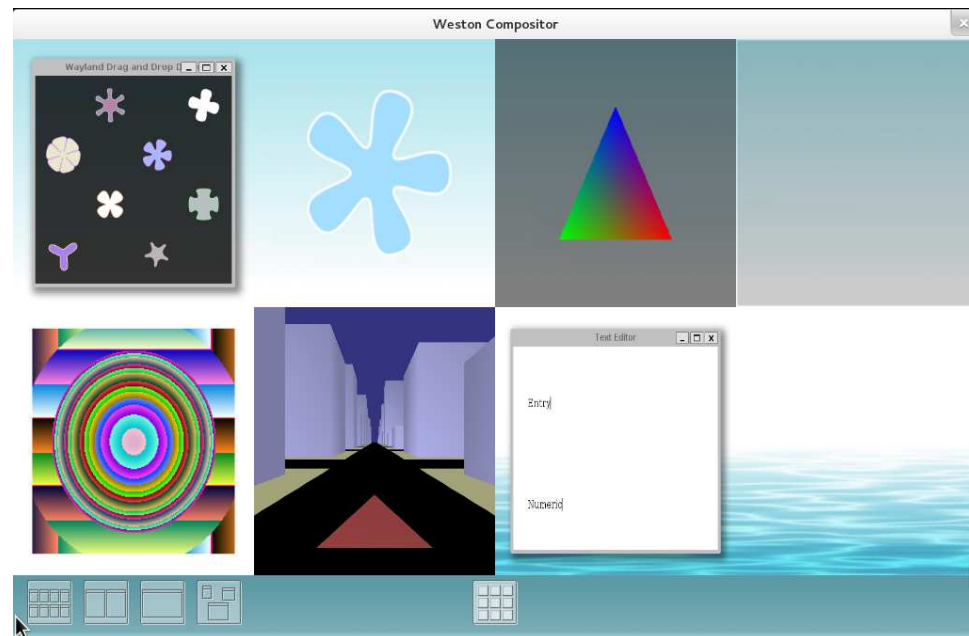
- **Any feedback is very Welcome in mailing list**
 - ◆ Benchmark
 - ◆ Collaborating other components e.g. Murphy
 - ◆ Etc.

■ Weston-ivi-shell-user-interface

- ◆ Ivi-application protocol support of sample application
 - Simple-egl/shm, editor, MockNavigation, and so on.
- ◆ Invoked by launcher

■ Reference of changing layout

- ◆ Internally ivi-layout APIs are used
- ◆ Transition animation



■ Wayland

- ◆ <http://wayland.freedesktop.org>
- ◆ <http://cgit.freedesktop.org/wayland>

■ GENIVI

- ◆ <http://projects.genivi.org/wayland-ivi-extension>