



Consolidation of IVI Graphic Subsystems Weston, a Wayland Compositor, and GENIVI Layer Management

Nobuhiko Tanibata
25 October 2013

Contents

- Trends of Graphic stacks in IVI segment
- Problems & Proposing solution
- Solution of consolidating two into one
- IVI shell protocol
- Design of Implementation
- Demonstration

Trend of Graphic stacks in automotive segment

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



- Trends
 - Tizen IVI
 - many companies shift proprietary stacks to using Wayland.

Problem & Proposing solution for selecting Wayland

- Problem: Two candidates of Wayland implementation. Which is better for automotive system?
 - Weston: a Wayland compositor
 - Genivi Layer manager: Wayland plug-in, a Wayland compositor
- Proposing Solution: Consolidate two into one. Merge both advantages. Why does DENSO do this?
 - DENSO is now developing new model product for several car makers, jointly implementing Wayland base automotive platform.
 - DENSO is motivated to open the implantation as product ready quality.

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
1.2.91	weston-1.2.91.zip weston-1.2.91.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

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

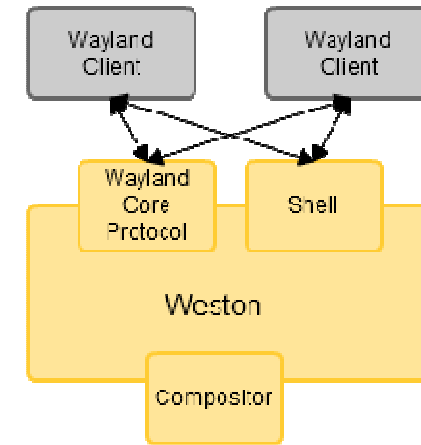
timo.lotterbach – 27 Mar 2013 – 12:51

IVI LayerManagement 1.0 release candidate 2 available

Weston: a Wayland compositor

Maintained by Wayland Community

- Wayland Core protocol
- Shell:
Managing SceneGraph
- Compositor:
Composite Surfaces



Pros:

- Wayland community is more active than Genivi Layer management, more contributors. Many bug fixes are available.

Cons:

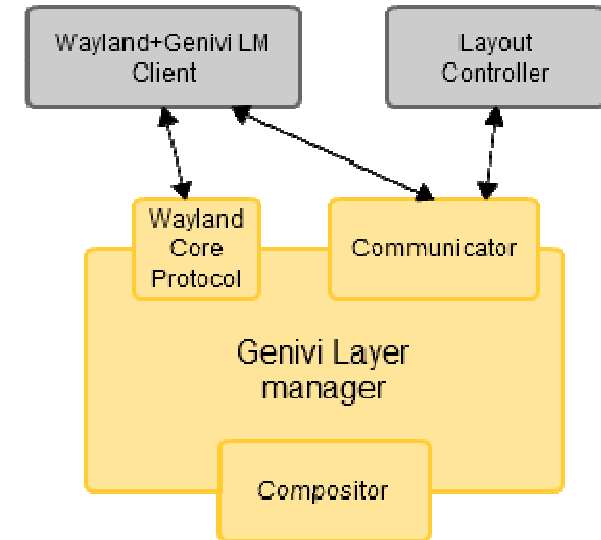
- No ivi feature set, Layer management.

GENIVI Layer management: Wayland plug-in



Maintained by GENIVI Layer management community.

- Wayland Core Protocol
- Communicator:
Genivi Layer management APIs
- Compositor:
Compositing Wayland surfaces
- **Pros:**
 - Support ivi feature set, Layer management
 - Car makers and Tier1s can use proven a HMI controller with Layer management. The controller is already qualified on actual product.
 - Denso contribute codes via Advance Driver Information Technology.
- **Cons:**
 - Compositor need to be updated continuously per Weston version up. E.g. Applying bug fixes.

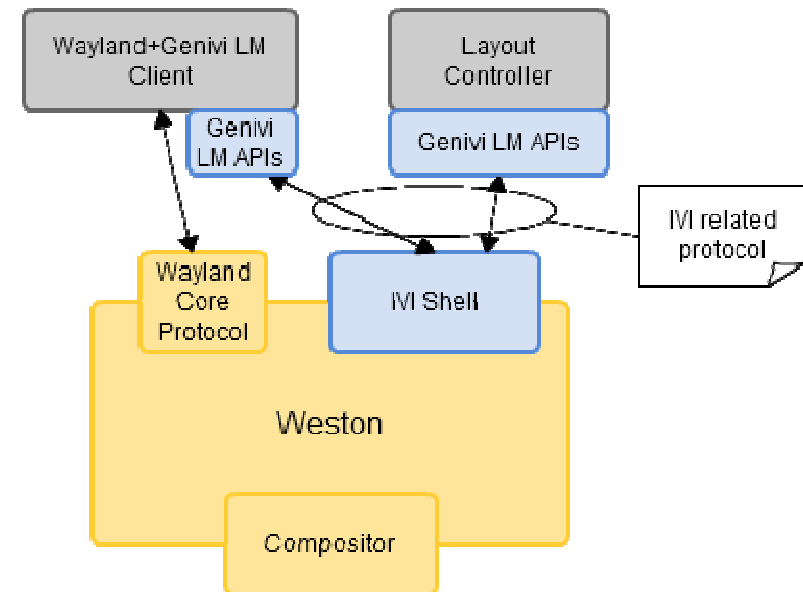


Solution of consolidating two into one

- Define IVI related protocol and IVI shell to Wayland community.
- Genivi Layer manage APIs wrapping IVI related protocol to Layer manager community.

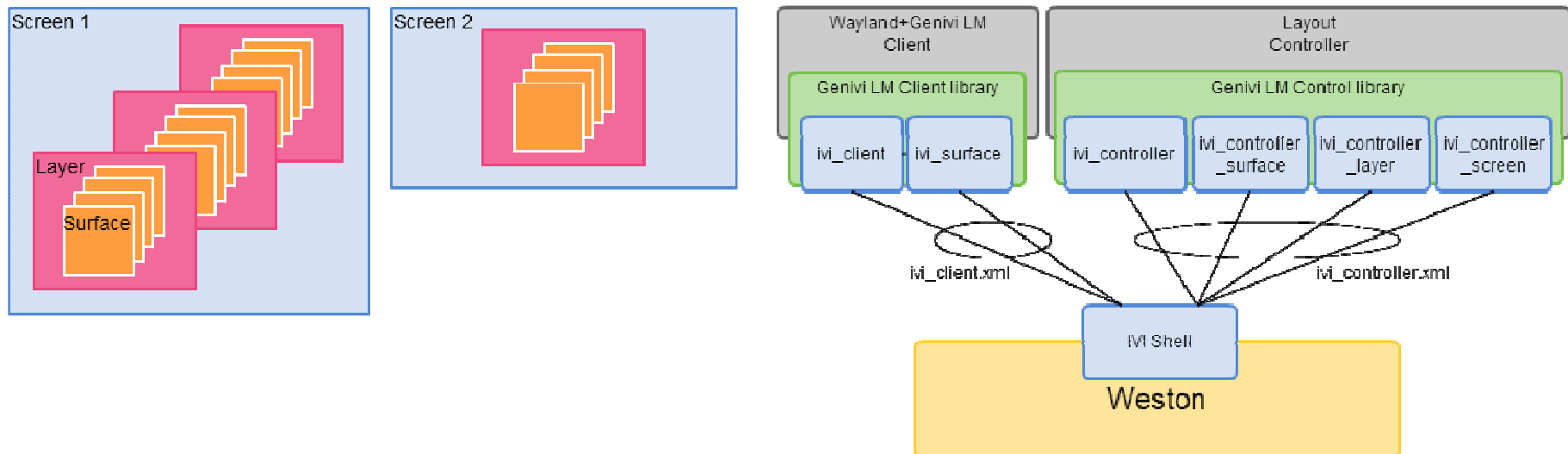
Comparison of IVI related features with current Wayland Protocol

GENIVI® Layer Management Client Interface Feature	Wayland Protocol
Surface Lifecycle Management	Yes
Surface Positioning	Yes, but through input device, not client supplied coordinate
Surface Visibility	No
Surface Opacity	No
Layer Lifecycle Management	No
Layer Positioning	No
Layer Visibility	No
Render order of surfaces in layer	No



IVI client/controller protocol

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



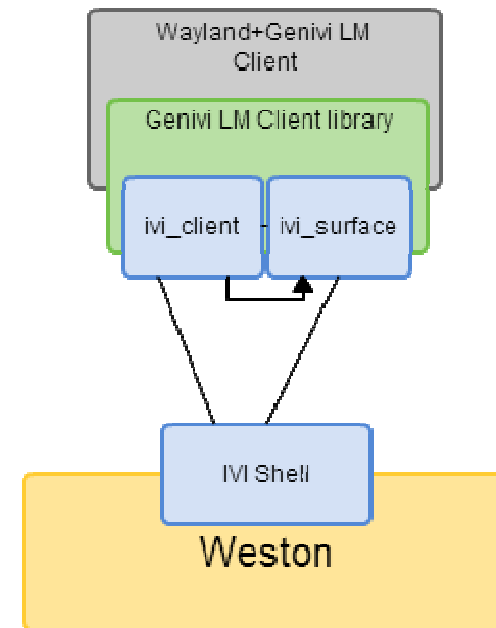
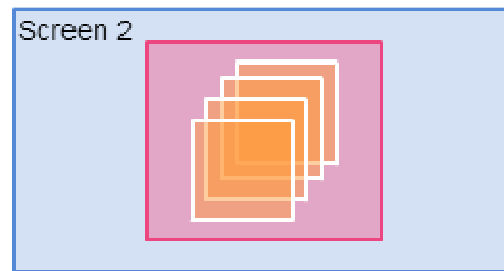
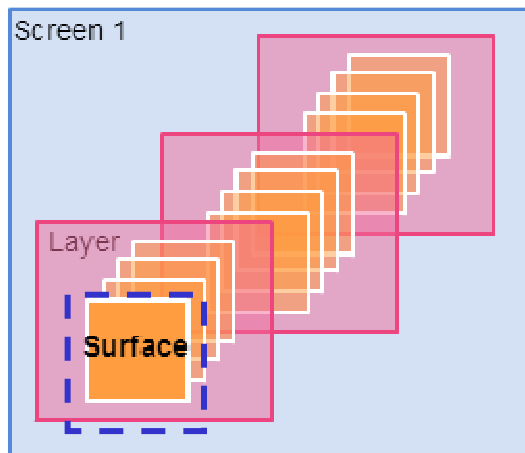
IVI Client protocol: ivi-client.xml

Use case: Wayland application set its native to ivi_surface

- ivi_client: 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.



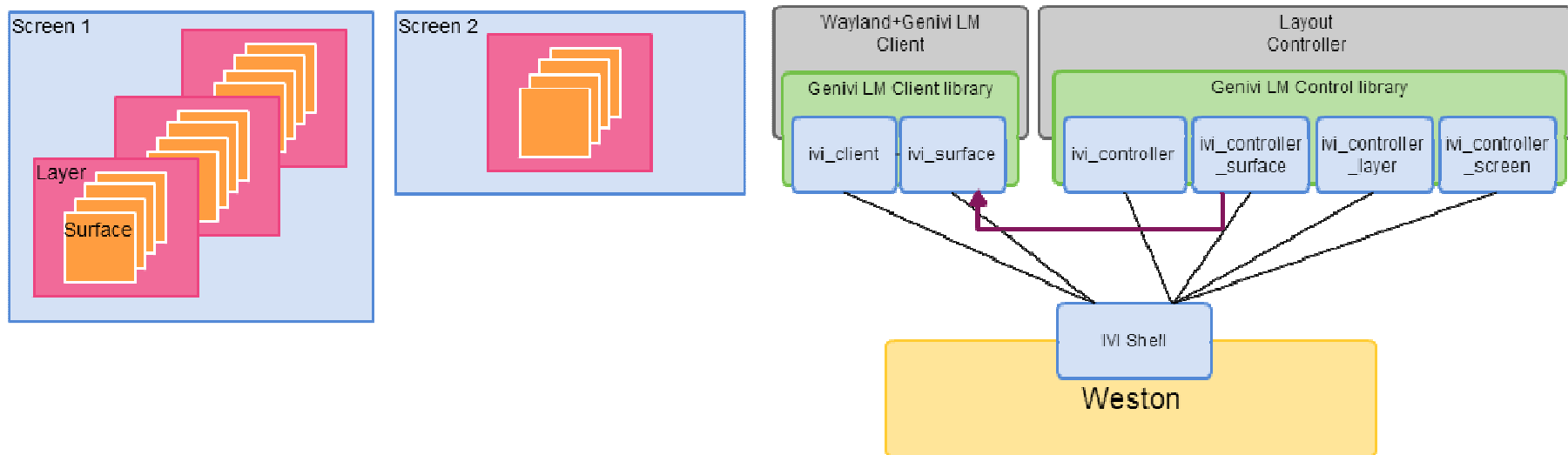
ivi-client.xml

```
<protocol name="ivi-client">
  . . .
  <interface name="ivi_client" version="1">
    <description summary="creation of ivi_surface"/>
    <request name="surface_create">
      <description summary="ilm_surfaceCreate"/>
      <arg name="id_surface" type="uint"/>
      <arg name="surface" type="object" interface="wl_surface" allow-null="true"/>
      <arg name="id" type="new_id" interface="ivi_surface"/>
    </request>
  </interface>
</protocol>
```

IVI controller protocol: ivi-controller.xml

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>

    ...
```

ivi-controller.xml: ivi_controller_layer

```
<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 wl_output"/>
```

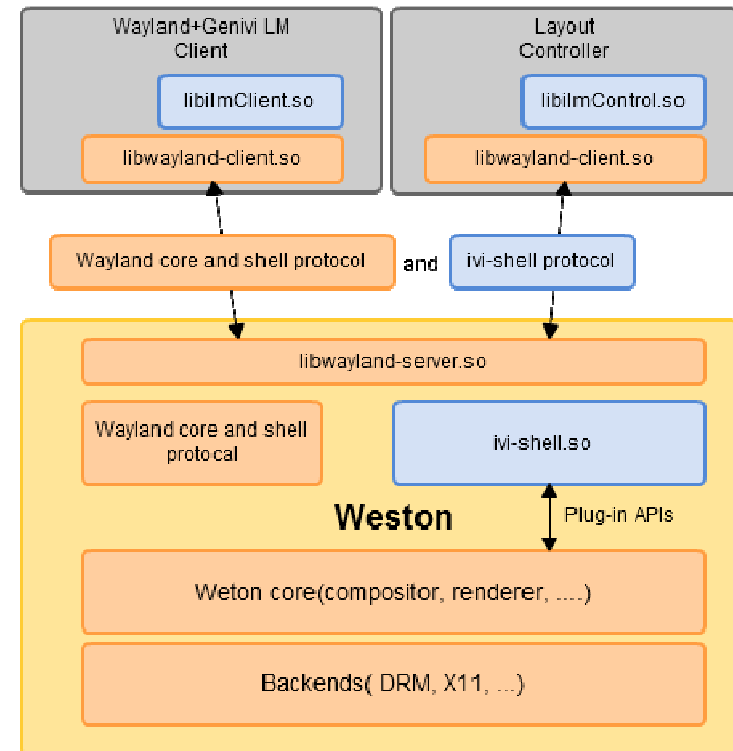
```
      <arg name="layer" type="object" interface="ivi_layer"/>
```

```
    </request>
```

```
    ...
```

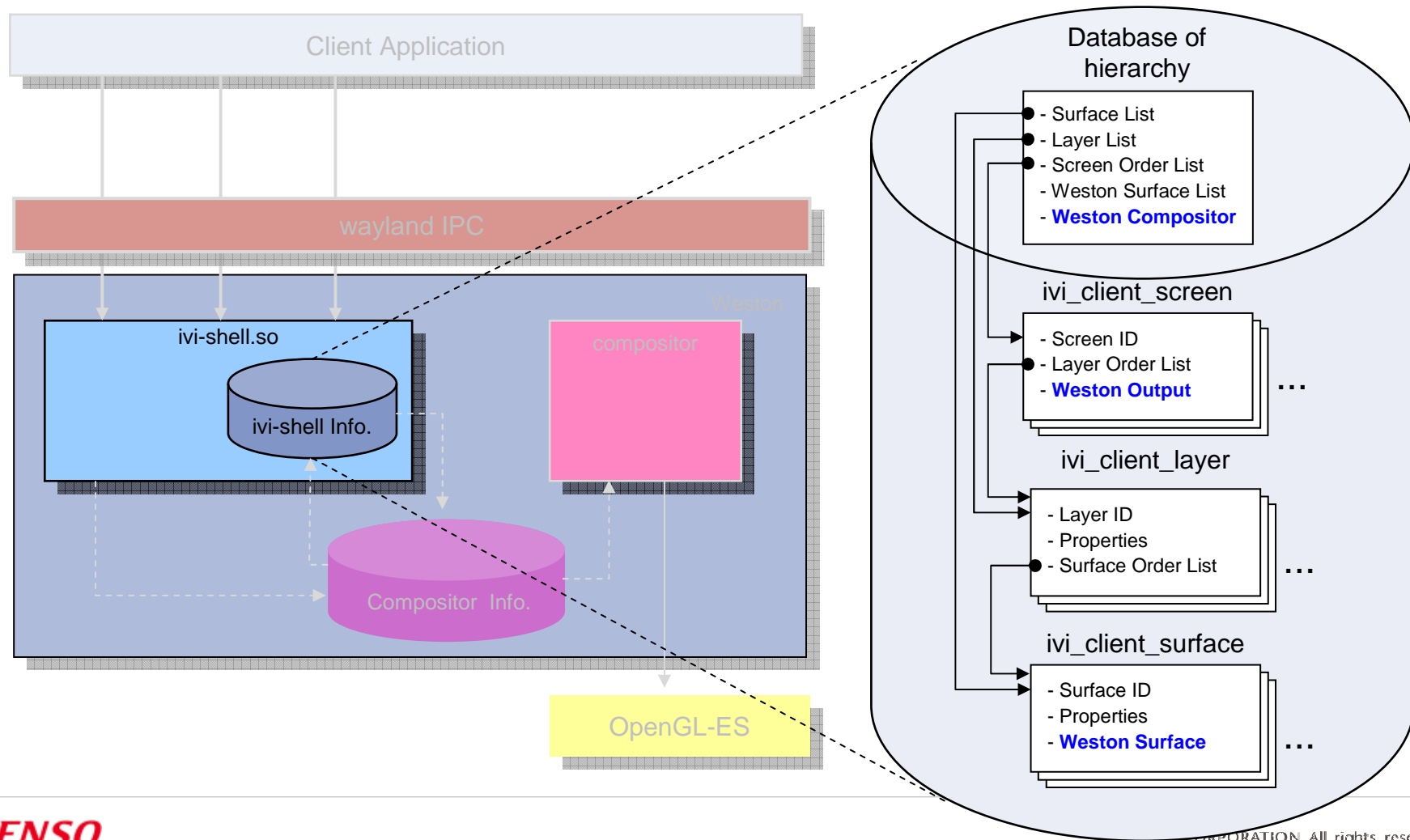
IVI-Shell Implementation

- **ivi-shell.so:**
 - Implement ivi-client/controller protocols
 - Communicate Weston by Plug-in APIs
 - Set name of plug-in in weston.ini
[shell]
type=ivi-shell.so
...
- **libilmClient.so**
 - Wrap ivi-client protocol to be compatible with Genivi LM client APIs.
- **libilmControl.so**
 - Wrap ivi-controller protocol to be compatible with Genivi LM control APIs



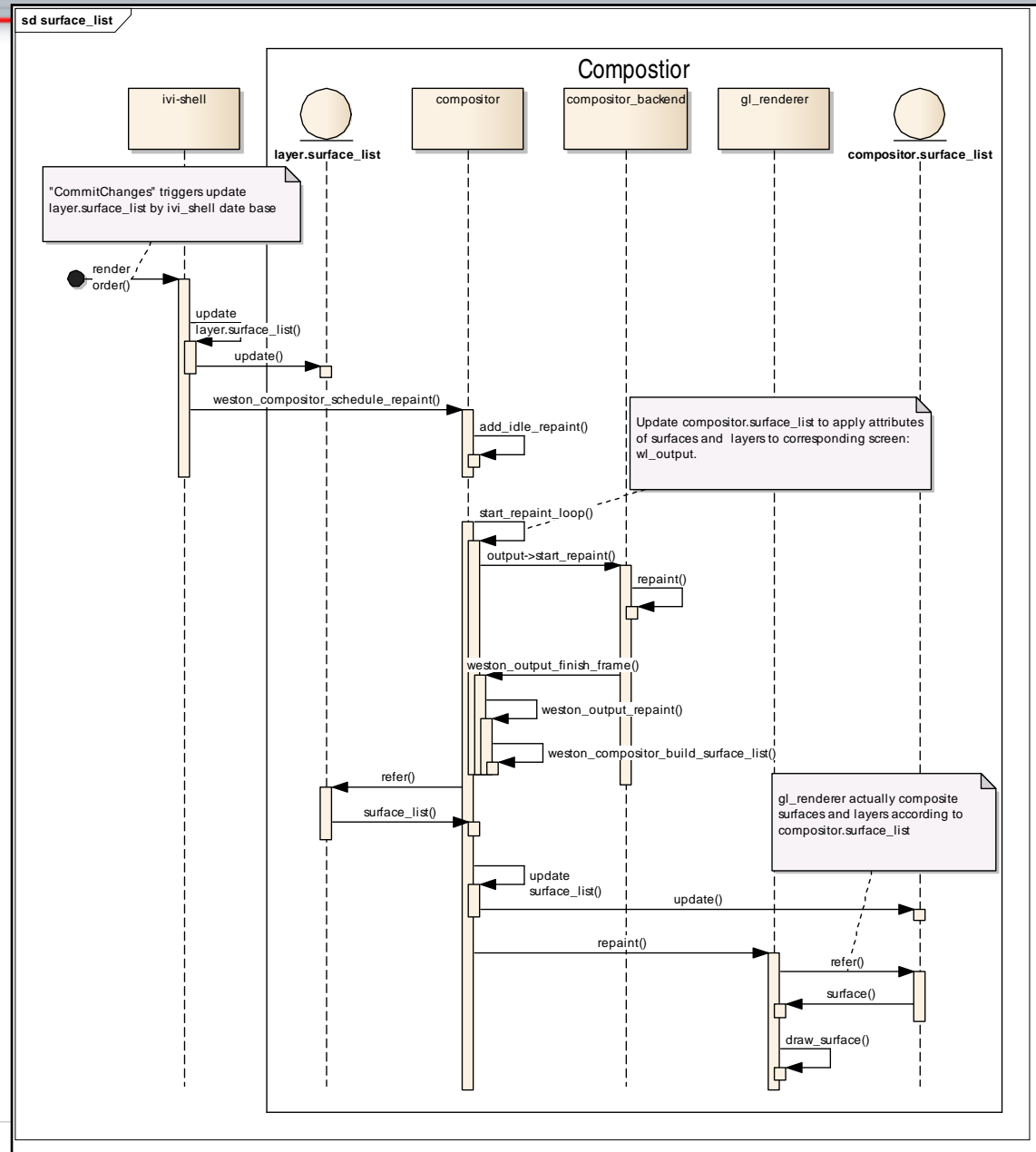
IVI-Shell property database

- ivi-shell.so contains a database to manage hierarchy and tie Weston resources.



IVI-Shell: how to composite surfaces

- IVI-shell just asks compositor to composite surface according to attributes.
- Update layer.surface_list by Database of attribute.



Demonstration

- Qt, EFL, and Adobe AIR on Wayland and Layout management by Genivi Layer management APIs.
- Layout change and fade-in animation done by compositor.

Weston Compositor - X1

FPS:48.6, MEM:19.3, DRW:3

Cluster on Adobe AIR

ADOBE AIR™

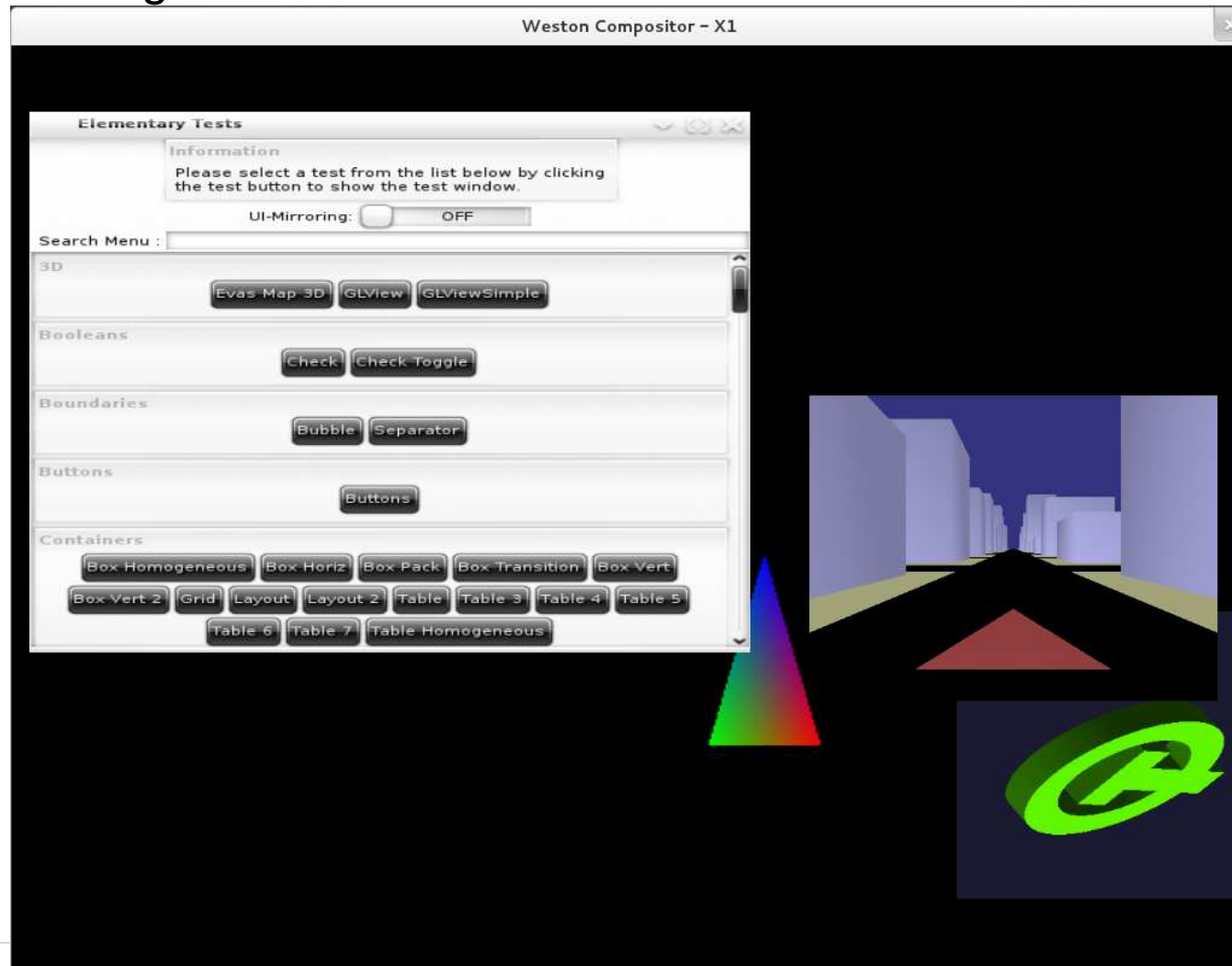
Qt and EFL windows

Qt

EFL

Demonstration

- Layer management compatible test
- LayerManagerControl demo



Summary

- Weston and Genivi Layer management wayland compositor have pros and cons
 - Propose consolidation of two compositor and being contributed to both community.
- IVI UI style; client and controller model
 - Define separated protocol
 - Genivi layer management compatible APIs

References

- Wayland
 - <http://wayland.freedesktop.org/>
 - <http://cgit.freedesktop.org/wayland>
- GENIVI
 - <http://projects.genivi.org/ivi-layer-management/>