

Experiences Developing a Wayland-Based Tizen IVI HMI

Ossama Othman

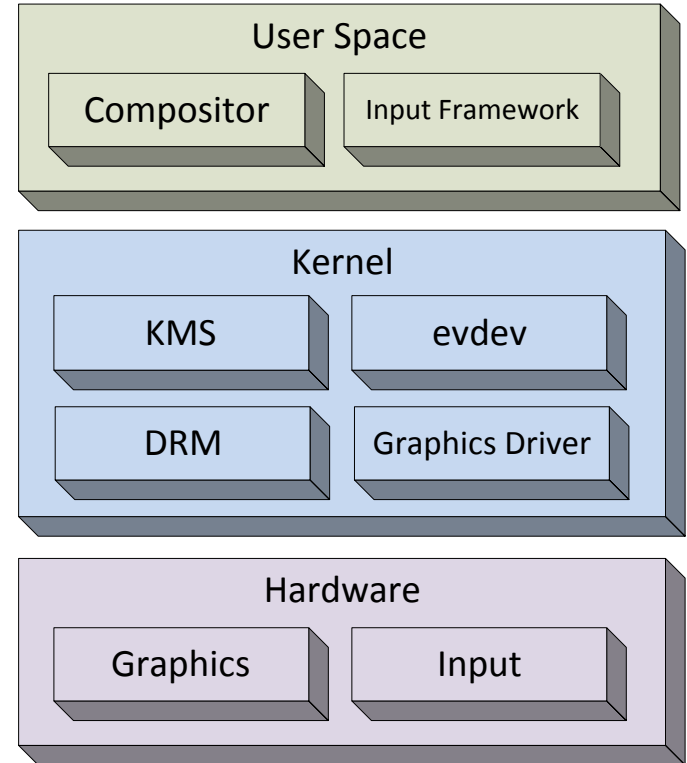
TIZEN[™]
**DEVELOPER
CONFERENCE**
2013
SAN FRANCISCO

Context

- **Provide human-machine interface (HMI) better suited for Tizen IVI**
 - Driver safety
 - React to vehicle state
- **Options**
 - Leverage existing user interfaces: mobile, desktop
 - Develop IVI-specific HMI

Problems


- **Mobile and desktop user interfaces**
 - Suboptimal screen geometry: resolution, orientation
 - Do not react to vehicle state
 - Not safe for drivers
 - Many features are not useful or suitable for IVI
 - Underlying graphics subsystem may be too large or slow
- **Develop IVI-specific HMI**
 - May require kernel and graphics hardware expertise
 - May require additional resources for in-house development
 - Cost



Solution

- **Develop IVI HMI by leveraging Wayland and HTML5/JS**
 - Efficient
 - Lightweight
 - Rapid prototyping
 - Simple architecture
 - Open-source
 - Avoids legacy issues with mobile and desktop





HMI Components

HMI Service

- Provides “home screen”
- Facilitates layout of application graphical surfaces
- **Out of scope**
 - Application lifecycle, sound and input management

Compositor

- **Handles interactions with graphics hardware**
- **Combines and renders multiple graphical “surfaces” into an image displayed on-screen**
- **Wayland**
 - Core compositor protocol used by IVI HMI
 - Additional IVI HMI-specific Wayland-based protocols to be defined
- **Weston**
 - Reference Wayland compositor
 - HMI will leverage extensibility to fill feature gaps

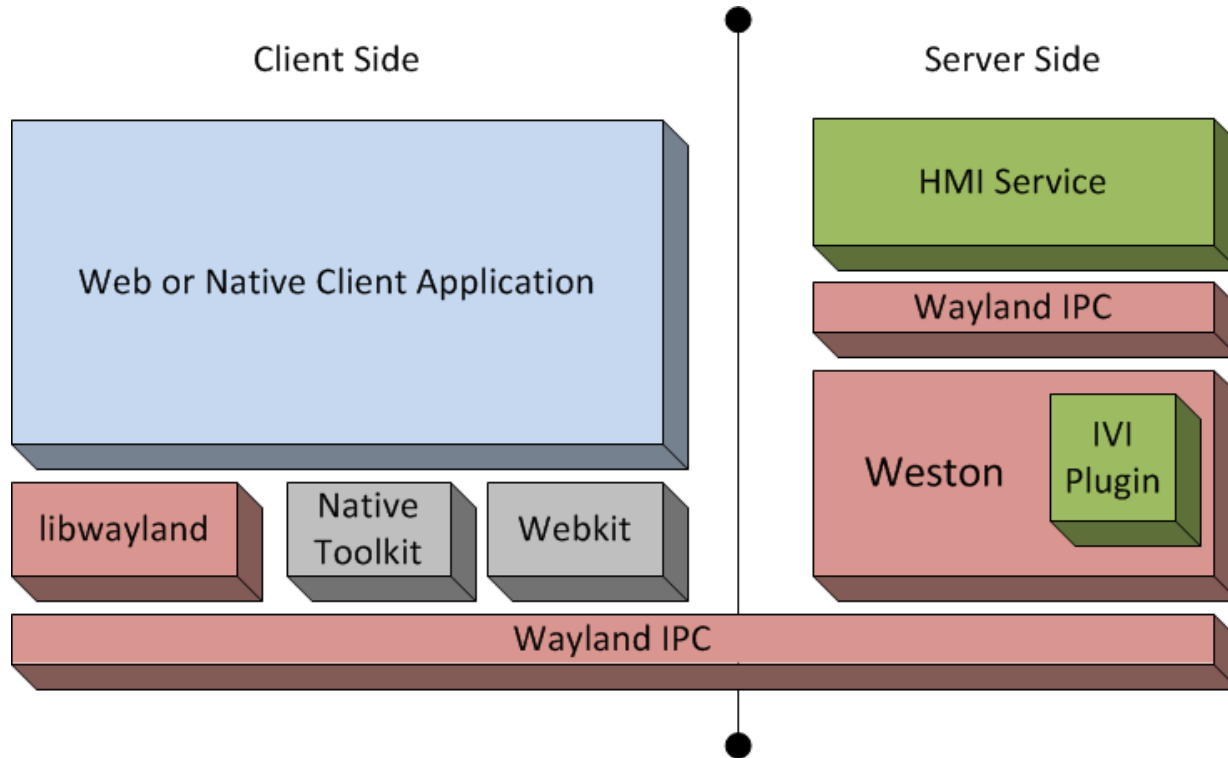
Application Run-time


- **Provides reusable set of functionality to applications**
- **HTML5**
 - Webkit
 - Web API implementations
 - Toolkits
- **Native**
 - C library
 - Middleware
 - Toolkits

Client Application

- **Draws surfaces in a buffer**
 - Cairo
 - OpenGL
 - etc
- **Shares buffer with compositor through Wayland**
- **Examples**
 - Media player
 - Browser

Component Interaction





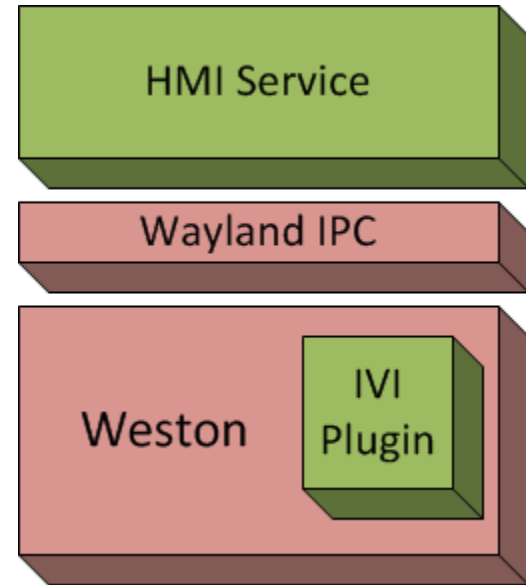
HMI Server Side

Server Side Architectures

- **HMI Service**
 - Stand-alone
 - Integrated
- **Server side architecture remains transparent to the client**
 - Just an implementation detail from the client point of view
 - Client does not determine where surfaces are placed

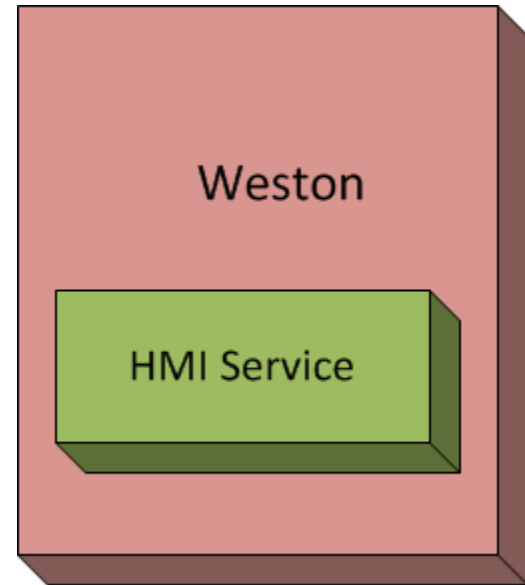
Stand-alone HMI Service

- **HMI service is another Wayland based client**
- **Separate process from compositor**
- **Communicates with compositor just as client application would**
- **Example: GENIVI style layer management**




Integrated HMI Service

- **HMI Service integrated with compositor**
 - Embedded directly in compositor plugin
 - Performs same functions as stand-alone service



HMI Server Side Architecture Comparison

Pros		Cons	
Stand-alone	Integrated	Stand-alone	Integrated
Reusable / Flexible	Simpler architecture	Flexibility may not be needed	Fatal error recovery may be more difficult
Better fatal error recovery	More efficient: less IPC	Less efficient: more IPC	Potentially easier to compromise compositor state
	Rapid prototyping		



HMI Server Side Problems

Problem: Layout of Both Web and Native Applications

- **HTML5 based HMI cannot readily place surfaces created by native application**
- **HMI is generally layer based**
 - Layer is a collection of surfaces
 - Sample layers: home screen decorations, input layer, split “screens”
- **No concept of layers in HTML5**
 - May be emulated using multiple HTML5 Canvases
- **Canvas based approach not always suitable**
 - Not all Web applications use Canvases
 - Cannot be used for native applications

Solution

- **Compositor**
 - Plugin exposes layer functionality required by HMI
- **Web run-time**
 - Plugin provides JavaScript API that allows interaction with the compositor
- **HMI alternatives**
 - Use WebSockets to communicate between compositor plugin and HTML5 based HMI
 - IVI-specific [Node.js](#) extensions

Other Problems

- **Direct use of `wl_shell` in application or toolkit**
 - Specific to desktop shell model / use case
 - Not entirely applicable to IVI
 - Compositor `wl_shell` implementation may not be available
 - Proposal for better handling of different Wayland shell types currently under review upstream

References

- **Tizen IVI Images**

- Daily: <http://download.tizen.org/releases/daily/2.0alpha/ivi-wayland/>
- Snapshots: <http://download.tizen.org/snapshots/2.0alpha/ivi-wayland/>
- Tizen 3.0 based IVI images are a work in progress

- **Wayland and Weston**

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

- **GENIVI®**

- <http://www.genivi.org/>
- <http://projects.genivi.org/ivi-layer-management/>

Legal

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY RELATING TO SALE AND/OR USE OF INTEL PRODUCTS, INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT, OR OTHER INTELLECTUAL PROPERTY RIGHT.

Intel may make changes to specifications, product descriptions, and plans at any time, without notice.

All dates provided are subject to change without notice.

Intel is a trademark of Intel Corporation in the U.S. and other countries.

*Other names and brands may be claimed as the property of others.

Copyright © 2013, Intel Corporation. All rights are protected.



TIZEN™

**DEVELOPER
CONFERENCE**

2013

SAN FRANCISCO