Efficient HMI Development for AGL
HI CORPORATION
Agenda

- AGL Render System Overview
- HMI Design in Automotive
- HMI Authoring Tool - Presentation of UI Conductor
- Bind HMI and Application Logic
- HMI Tests in Automotive
AGL Render System: Wayland protocol

Client 1 Application

Surface 1 (Video memory buffer)

Request update

Wayland Compositor (RI: Weston)

Display

Client 2 Application

Surface 2 (Video memory buffer)

Request update
Rendering from client application

- No rendering API in Wayland
- Direct, client side rendering
  - Direct call OpenGL|ES / OpenVG / Software renderer / …
  - GTK
  - Qt
  - …

Choose the one that fits your needs.
HMI Design in Automotive

Need more than just rendering things.

- Separate specifications/planifications from design
Separate Specifications from Design

In the Automotive world, HMI is often developed by Tier1 companies for the OEM.

Specifications by the OEM or the Tier1.
- Visual presentation
- Early prototyping

- Planification by engineers / planers
- Design by designers
  - Build interactions
  - Animate the GUIs
HMI Design in Automotive

More than just rendering things.

- Separate specifications/plan from design

- Test usability and functionalities from the start

- Bind HMI with the vehicle (Hardware, CAN, State Machine)

Using only a render library will take a lot of time.

Frameworks and authoring tools help reducing production time.
UI Conductor

Embedded GUI library

Authoring Tool
Improved Development Process

Before

- Planner
- Designer
- Programmer

Create temporary data → Production data → Check → Fix → Fix → Fix

First check will happen at late development phase.

After

- Planner
- Designer
- Programmer

Create data → Fix

You can check while creating the data.

Programming and implement to real device → Aggregate to UI Conductor. Check in the tool.
**UI Conductor – Flow**

**Architect / Design**

- ConductorTact
  - **Event, Layout, Screen Transitions, Animations**
  - **Binary files** `.nbr`, `.nbu`
  - **Document output** `.xlsx`
  - Conductor Runtime
    - **Library built for the targeted platform**
    - **Link**

**Program**

- **Customer application**
  - **Use Public API**

**Design**

- **Import/Export plugin**
  - **Layout info**
    - .PNG
    - .XML

(C)2016 HI CORPORATION. All Rights Reserved.
Authoring demo with UI Conductor Tact
How to bind  HMI and Application Logic

Building HMI is not only “Appearance”.

Bind HMI with application logic
Organization HMI application

Model View Controller model (MVC)

1. View
Screen layout, design, animations, … visible components.

2. Model
Data that must be shown to the user.
Contains data from the environment (CAN, etc.)

3. Controller
Binds the Model to the View(s)
Manages the update of the model from interactions with the view.
Manages the update of the view after modification of the model
State machines (as a model)

State machine should not be handled by the UI.

Lightweight binding is preferable.
Binding with a state machine

Example:
1: Press HW button to change something on the screen (in red)
2: Press a UI button on the screen and update the state machine (in blue)

※The State Machine implementation does not matter.
Binding with State Machine demo

This demo has been done with ZIPC
(ZIPC is a state machine software made by CATS)

But you can use any other state machine library.
HMI tests in Automotive

In Automotive, the final target is the vehicle.

Want to check Usability, Ease of use, Security.

Three approaches:
- Embed the program in the vehicle and test
- Bind a pc / tablet to the vehicle input interfaces and test
- Emulate the vehicle input data and test
• Binding with CAN Demo

• Binding with input interfaces Demo
Bind HMI to your application

Create surface with Wayland and pass it

Forward events

Handle state machine if needed and notify the HMI of the state changes

And your GUI is working on your target.
Thank you!