Agenda

- Window Manager
  - Requirements
  - Plan

- Window manager Features
  - Overview architecture
  - Internal components
  - Features
  - Use cases (sample)
  - Sequence chart (draft)

- Window manager APIs (draft)
Window Manager

Requirements

- To support multi applications and multi GUI (e.g., Qt, HTML5, JavaFX, EB)
- To support (OEM specific) Policy
- To support multi ECU and Multi Display
AGL HMI FW development – TMC CY2017 roadmap

<table>
<thead>
<tr>
<th>AGL activity</th>
<th>2017 - 1st Half</th>
<th>2017- 2nd Half</th>
</tr>
</thead>
<tbody>
<tr>
<td>15 May HMI-FW Draft</td>
<td>15 May HMI-FW Draft</td>
<td>15 Sep EE RC1</td>
</tr>
<tr>
<td>1 Jun ALS2017 Tokyo</td>
<td>11-13 July AGL F2F US</td>
<td>1 Oct ANM Munich</td>
</tr>
<tr>
<td>1 July</td>
<td>1 Aug</td>
<td>15/Nov EE RC3</td>
</tr>
<tr>
<td>11 July</td>
<td>14 Aug</td>
<td>#/Dec EE v5.0.0</td>
</tr>
<tr>
<td>1 Sep</td>
<td>15 Sep</td>
<td>8/Jan CES2018</td>
</tr>
<tr>
<td>15/Sept EE RC1</td>
<td>1/Oct ANM Munich</td>
<td></td>
</tr>
<tr>
<td>1/Oct EE RC2</td>
<td>#/Nov EE RC3</td>
<td></td>
</tr>
<tr>
<td>1 Dec</td>
<td>#/Dec EE v5.0.0</td>
<td></td>
</tr>
<tr>
<td>1 Dec</td>
<td>8/Jan CES2018</td>
<td></td>
</tr>
</tbody>
</table>

**Window Manager Development (Mentor)**
- WM1: Decide Functionality
- WM3: Share APIs to AGL
- WM4: First Demo
- WM5: Complete WM (Server+Client)

**Sound Manager PoC Development (Mentor)**
- SM1: Decide Functionality
- SM3: Share APIs to AGL
- SM4: First Demo
- SM5: SM PoC (Server+Client)

**HomeScreen (Nexty)**
- Start Design Based on WM3.
- Start Implementation based on WM3 and WM4.

**APP development (External App Developers)**
- Start Modifying APPs based on WM4.

**Legend:**
- Critical Milestones
- General Milestones
- Completed
- TBD
- Dependency

HMI FW presentation with SM PoC demo & WM Demo in AGL AMM Munich by TMC
Window Manager – Overview 1/3

Architecture

Legend:
- Window Manager
- APP
- Wayland
Window Manager – Overview 2/3

Internal Components and External dependencies

Legend:
- Window Manager
- OEM Specific
- APPs
- Wayland

Qt Application
- API Client Library

Java FX Application
- API Client Library

HTML Application
- API Client Library

EB Application
- API Client Library

Weston
- Wayland Compositor
- IVI Shell

Window Manager
- API Service
- Resource Manager
- Policy Manager
- Layout Manager

Wayland Interface
- Wayland Client

Resource Database

Policy Database

Layout Database
Window Manager – Overview 2/3

API References:

1. APP → WM
   request to generate surface-id

2. WM → APP
   WM returns surface-id to APP

3. APP → Weston
   create surface

4. WM → Weston
   register surface to be able to control it later
   registerSurface(surfaceID),
   configure(surfaceID, geometry)

5. Weston → WM
   acknowledge

6. WM → APP
   Visible, Invisible, Active, Inactive

※ Agreed with The QT Company
Window Manager – Sample Use Cases

1. Full-screen -> Full-screen
2. Full-screen -> Split-screen
3. Split-screen -> Pop-up (on-screen)
4. Split-screen -> Full-screen
Window Manager – Sample Use Cases 1/4
Window Manager – Sample Use Cases 2/4
Window Manager – Sample Use Cases 3/4

Two Applications SplitScreen to Popup FullScreen Sequence Diagram

FirstApp
SecondApp
PopupApp
WindowManager
Weston

requestApplication(AppID3)
allocateWindowResource(AppID3, areaID3_popup)

PolicyManager::checkPolicy()
LayoutManager::checkLayout()
configure(areaID3_popup, geometry)

updateSurface()
flushDraw()
paint()
swapBuffers()

inactive()
inactive()
active()
invisible()
invisible()
visible()
syncDraw()
endDraw()
Window Manager – Sample Use Cases 4/4
# Window Manager – APIs

<table>
<thead>
<tr>
<th>S. No.</th>
<th>API</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>bool registerApplication(int appid)</td>
<td>Register one application with id appid, tied to the PID determined by the connection.</td>
</tr>
<tr>
<td>2</td>
<td>int areaid registerSurface(int appid, surface_type type)</td>
<td>Provide the areaid to an application to then create its owned surface with id areaid and type. Type can be something like default (generic surface, no special handling) or popup (which must be placed on a popup-layer).</td>
</tr>
<tr>
<td>3</td>
<td>bool allocateWindowResources(int appid, int areaid)</td>
<td>Activate the areaid application surface, i.e. make it visible in its assigned area according to layout</td>
</tr>
<tr>
<td>4</td>
<td>bool deallocateWindowResources(int appid, int areaid)</td>
<td>Deactivate an application surface, based on the areaid</td>
</tr>
<tr>
<td>5</td>
<td>void endDraw(int areaid)</td>
<td>Rendering Completed</td>
</tr>
<tr>
<td>6</td>
<td>bool requestApplication(int appid)</td>
<td>Request to window manager for application with appid to be displayed, this might trigger a layout change, depending on policies and current layout.</td>
</tr>
</tbody>
</table>
Window Manager – APIs

<table>
<thead>
<tr>
<th>S. No.</th>
<th>API</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><code>void visible(int areaid)</code></td>
<td>Notify that area areaid has become visible</td>
</tr>
<tr>
<td>2</td>
<td><code>void invisible(int areaid)</code></td>
<td>Notify that area areaid has become invisible</td>
</tr>
<tr>
<td>3</td>
<td><code>void active(int areaid)</code></td>
<td>Notify that an area areaid is currently active (has focus)</td>
</tr>
<tr>
<td>4</td>
<td><code>void inactive(int areaid)</code></td>
<td>Notify that an area areaid is currently inactive (has lost focus)</td>
</tr>
<tr>
<td>5</td>
<td><code>void layoutChanged(int areaid, layout newlayout)</code></td>
<td>Notify a client area that its layout changed</td>
</tr>
<tr>
<td>6</td>
<td><code>void syncDraw(int areaid)</code></td>
<td>Redraw area after layout change</td>
</tr>
<tr>
<td>7</td>
<td><code>void flushDraw(int areaid)</code></td>
<td>Notify client that it should swap buffers (after SyncDraw)</td>
</tr>
<tr>
<td>8</td>
<td><code>void endDraw(int areaid)</code></td>
<td>Notify client that the drawing process ended after swap buffers</td>
</tr>
<tr>
<td>9</td>
<td><code>void popupTimedOut(int areaid)</code></td>
<td>Notify a client, that its popup surface timeout is expired</td>
</tr>
</tbody>
</table>
THANK YOU
REFERENCES
Window Manager

1. Area Vs Surface

2. Resource Management
Window Manager

- Visible
- Invisible
- Active
- Inactive