developers’ prove of concept demo
Agenda

- Wayland introduction
- Embracing wayland for tizen
- Performance enhancement
Wayland Introduction
Usage scope
What is wayland

- Protocols

- Wayland is a protocol for a compositor to talk to its clients as well as a C library implementation of that protocol. (Kristensen, Kristian)

- Weston is one compositor (Kristensen, Kristian)
Xorg & Wayland architecture

XClient

X Server

Compositor

Wayland Client

Wayland Compositor

Less IPC In wayland
How does wayland/weston work?

Weston-compositor

- drm backend
- Customized backend

Wayland protocols

- shell
- shell client (wayland-client)

Other Wayland clients
Embracing wayland for Tizen
Tizen graphics stack with Xorg
Tizen graphics stack with Xorg

- e17 (Compositor)
- Multimedia/gstreamer
- Native UI FW
- Web UI FW
- WebKit
- Cairo
- Freetype & Fontsconfig
- edje
- ecore
- eet
- evas
- eina
- Xorg
- EGL
- Open GL ES
- pixman
- Linux Kernel
- DRM
- GPU Driver
- G2D Driver
- HW
Tizen graphics stack with Wayland

Native Apps
- Weston/e17 (Compositor)
- Multimedia/gstreamer
- Native UI FW

Web Apps
- Web UI FW
- WebKit

Native UI FW
- elementary
  - edje
  - ecore
  - eet
  - eina
  - evas
  - ...

Wayland protocols
- Open GL ES
- EGL
- pixman

- Cairo
- Freetype & Fontsconfig

Linux Kernel
- DRM
- GPU Driver
- G2D Driver

HW
Tizen graphics stack with Wayland

Native Apps
- Weston/e17 (Compositor)
- Multimedia/gstreamer
- Native UI FW
- elementary
  - edje
  - ecore
  - eet
  - evas
  - eina
  - ...

Web Apps
- Web UI FW
- WebKit
- Cairo
  - Freetype & Fontsconfig
- pixman

Wayland protocols
- EGL
- Open GL ES
- pixman

Linux Kernel
- DRM
- GPU Driver
- G2D Driver

HW
Changes in general

- Remove hardcoded Xorg dependency
- Add wayland protocol
- Extend EGL
- Add compositor
- Upgrade EFL
- Tizen Frameworks porting to Wayland
  - App FW, Multimedia FW, Webkit/WRT…
Weston compositor

- Weston
- Drm backend
- tizen Shell
- HOME UX

KMS | Wayland server | Wayland clients

gles+egl
Weston compositor

HOME UX
  Menuscreen
  + Indicator
  + Lock screen
  ...

KMS  Wayland server  Wayland clients

Drm backend

weston  tizen Shell

gles+egl

Intel Software Professionals Conference - Intel Confidential
EFL with Wayland support

APP

elementary

Ecore Evas

wayland engine

ecore wayland

EGL

surface commit

frame done

input event

weston
Native Applications

• **EFL**
  • Add the patches from upstream

• **APPFW**
  • Hide/show/Rotation to be re-implemented

• **Remove X Related API dependencies**
Webkit2

- Buffer sharing between web and browser process
  - wl surface (with dummy wayland egl window) to fake X pixmap
MultiMedia (with libva)

- Driver render to wayland buffer
- Libva wayland backend
  - setting up bridge between server and client
- Gstreamer vaapi video sink
  - Attach wl_buffer to wl_surface
Video driver
Libva wayland backend

wl_buffer

Video driver
Libva wayland backend

wl_buffer

attach

wl_surface

gstreamer vaapi

Video driver
gstamer vaapi

wl_surface
Benefits

• Memory saving in video
  • Flexible buffer type (RGB/YUV), direction and size for composition
  • Inherent all benefits for overlay
• Thin architecture for performance tuning
Performance

Wayland’s thin architecture makes it possible and easier
Performance

- **Frame Rate**
  - 60 FPS (Frames Per Second)
  - 16ms for one frame from client to compositor
Performance: tool

- **E-Graph**
  
  We developed a tool to visualize log information and draw FPS curve.  
  
  Open source project hosted at [https://gitorious.org/e-graph/e-graph](https://gitorious.org/e-graph/e-graph)
Live Demo for E-graph
Original state

- Fps curve and timing of critical events (drawn by E-Graph)
- For the scroll animation for org.tizen.Settings
The famous Triple buffering

- Add one buffer for the client and compositor to draw when the resource is blocked by waiting VSync
Triple buffering

• *Before*
  ~40fps

• *After*
  ~48fps
Triple buffering

- **Before**
  - ~40fps

- **After**
  - ~48fps

Far away from 60fps
GPU usage

Time spend on composition

\(~13\text{ms!}\)
Opaque region

- Weston needs opaque region information to do more efficient compositing
Root cause

- No opaque region set for surface
- Weston redraw the overlapped surface
- Heavy work load during composition

Action

to set the opaque region for wayland surface in ecore
Opaque region

- Compositing time
  ~13ms → ~5ms

- FPS
  40fps → 60fps
Embracing wayland

• Prove of Concept result: it’s Doable
• Wayland brings thin architecture for compositor and clients
• Easier to get to the performance goal
Wayland Upstream Resources

- Maillist: wayland-devel@lists.freedesktop.org
- Wiki Page: wayland.freedesktop.org
- E-graph: https://gitorious.org/e-graph/e-graph