Utilizing Wayland to Improve Performance on Tizen
运用Wayland提升在Tizen平台的表现

Derek Foreman / Christopher Michael
Samsung
Overview

X
Wayland (and Weston)
Under the hood differences
EFL on X
EFL on Wayland
Enlightenment: X vs Wayland
Missing features:
Missing features:
Compositing
Missing features:
Compositing
3D Graphics
Missing features:
Compositing
3D Graphics
Video
Missing features:
Compositing
3D Graphics
Video
Moving windows around
X – with Extensions

- MIT-SHM
- GLX
- Xv
- Composite
- 24 More!

Missing features:
- Compositing
- 3D Graphics
- Video
- Moving windows around
X – with Extensions and Support

MIT-SHM
GLX
Xv
Composite
24 More!

Missing features:
- Compositing
- 3D Graphics
- Video
- Moving windows around

libGL (via GLX)

DRM/KMS
EVDEV
X – with Extensions and Support and Clients

MIT-SHM
GLX
Xv
Composite
24 More!

X

Compositor
Window Manager
Application
libGL (via GLX)

DRM/KMS
EVDEV
Wayland

Missing features:
Everything

Application

GL (via EGL)

DRM/KMS

EVDEV
Wayland and Weston

- DRM/KMS
- EVDEV

Weston

Wayland

Application

GL (via EGL)
Under the Hood: Architecture

X

X Server
- Talks to hardware
- Does rendering
- Sends events
Under the Hood: Architecture

X

Window Manager
Moves windows
Handles some state
Does rendering

X Server
Talks to hardware
Does rendering
Sends events
Under the Hood: Architecture

X

Compositor
Ensures tear free display
Wobbly Windows
Does rendering

Window Manager
Moves windows
Handles some state
Does rendering

X Server
Talks to hardware
Does rendering
Under the Hood: Architecture

X

Compositor *
Ensures tear free display
Wobbly Windows
Does rendering

Window Manager *
Moves windows
Handles some state
Does rendering

X Server
Talks to hardware
Does rendering
Sends events

* May or may not be the same program
Under the Hood: Architecture

X
X Server
Window Manager
Compositor

Wayland
Compositor
<table>
<thead>
<tr>
<th>X</th>
<th>Wayland</th>
</tr>
</thead>
<tbody>
<tr>
<td>X Server</td>
<td>Compositor (Does all that stuff)</td>
</tr>
<tr>
<td>Window Manager</td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>Wayland</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>X Server</td>
<td>Compositor</td>
</tr>
<tr>
<td>Window Manager</td>
<td></td>
</tr>
<tr>
<td>Compositor</td>
<td></td>
</tr>
</tbody>
</table>

Under X, compositor not involved with input
Benefits to Tizen:

Compositor puts windows anywhere any orientation input works
Under the Hood: Wayland multi-seat

Collection of input devices:
  Keyboard
  Mouse
  Touch
Under the Hood: Wayland multi-seat

Collection of input devices:
   Keyboard
   Mouse
   Touch

Associated state:
   Cut and paste
   Focus
   Interactive moves and resizes
Under the Hood: Wayland multi-seat

Benefits to Tizen:

Effective sharing of displays
New collaborative possibilities
<table>
<thead>
<tr>
<th>Under the Hood: Drawing API</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
</tr>
<tr>
<td>Rectangles</td>
</tr>
<tr>
<td>Lines</td>
</tr>
<tr>
<td>Arcs</td>
</tr>
<tr>
<td>Filled Polygons</td>
</tr>
<tr>
<td>Filled Arcs</td>
</tr>
<tr>
<td>Fonts</td>
</tr>
<tr>
<td>Image buffers</td>
</tr>
<tr>
<td>Wayland</td>
</tr>
<tr>
<td>Image buffers</td>
</tr>
</tbody>
</table>
Under the Hood: Drawing API

X Junk
Image buffers

Wayland Image buffers
Benefits to Tizen:

Less completely useless code
Saves storage space and memory
<table>
<thead>
<tr>
<th>Client draws stuff</th>
<th>Client draws stuff</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client sends buffer</td>
<td>Client commits buffer handle</td>
</tr>
<tr>
<td>X server does stuff</td>
<td>Compositor renders</td>
</tr>
<tr>
<td>X server alerts compositor</td>
<td>Compositor releases buffers</td>
</tr>
</tbody>
</table>
| Compositor renders | }
Under the Hood: Drawing API

X

Client draws stuff
Client sends buffer
X server does stuff
X server alerts compositor
Compositor renders

X server is single threaded

Wayland

Client draws stuff
Client commits buffer handle
Compositor renders
Compositor releases buffers
### Under the Hood: Drawing API

<table>
<thead>
<tr>
<th>X</th>
<th>Wayland</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client draws stuff</td>
<td>Client draws stuff</td>
</tr>
<tr>
<td>Client sends buffer</td>
<td>Client commits buffer handle</td>
</tr>
<tr>
<td>X server does stuff</td>
<td>Compositor renders</td>
</tr>
<tr>
<td>X server alerts compositor</td>
<td>Compositor releases buffers</td>
</tr>
<tr>
<td>Compositor renders</td>
<td></td>
</tr>
</tbody>
</table>

- X server is single threaded
- Handles client storage
<table>
<thead>
<tr>
<th>X</th>
<th>Wayland</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client draws stuff</td>
<td>Client draws stuff</td>
</tr>
<tr>
<td>Client sends buffer</td>
<td>Client commits buffer handle</td>
</tr>
<tr>
<td>X server does stuff</td>
<td>Compositor renders</td>
</tr>
<tr>
<td>X server alerts</td>
<td>Compositor releases buffers</td>
</tr>
<tr>
<td>Compositor renders</td>
<td></td>
</tr>
</tbody>
</table>

Using poorly synchronized client data
Under the Hood: Drawing API

X

Client draws stuff
Client sends buffer
X server does stuff
X server alerts compositor
Compositor renders

Wayland

Client draws stuff
Client commits buffer handle
Compositor renders
Compositor releases buffers

Using poorly synchronized client data
Tear-free display possible… sometimes
Benefits to Tizen:

“Every frame is perfect”
Easier benchmarking and profiling
Lower impact from a bad application
Under the Hood: Wayland buffer properties

Buffer scale
   Integer multiple
   Compensates for display DPI difference
Under the Hood: Wayland buffer properties

Buffer scale
- Integer multiple
- Compensates for display DPI difference

Transform
- 90 degree rotations
- Application can render in display orientation
Under the Hood: Buffer Properties

Benefits to Tizen:

Matching transforms may save copy
Scale allows for DPI awareness
Benefits to Tizen:

Matching transforms may save copy
Scale allows for DPI awareness

Less cpu usage, better images
Wayland provides a frame callback to clients
Means “commit now to hit vblank”
Doesn’t happen for hidden apps
Or sleeping displays
Under the Hood: Frame callbacks

Benefits to Tizen:

Reliable timing mechanism
Simple, low latency
Under the Hood: Frame callbacks

Benefits to Tizen:

- Reliable timing mechanism
- Simple, low latency
- Smooth and responsive animations
<table>
<thead>
<tr>
<th>Xv</th>
<th>Wayland</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenGL</td>
<td>Buffers (fourcc)</td>
</tr>
<tr>
<td>Image</td>
<td>Alpha Hole</td>
</tr>
</tbody>
</table>
Under the Hood: Video Handling

- X
- Xv
- OpenGL
- Image
- Alpha Hole

- Wayland
  - Buffers (fourcc)

"Descriptive, not prescriptive"
Benefits to Tizen:

- Compositor can leverage hardware planes
- Less cpu usage
- Better quality video
- Longer battery life
## Under the Hood: Video Handling

<table>
<thead>
<tr>
<th>X</th>
<th>Wayland</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xv</td>
<td>Buffers (fourcc)</td>
</tr>
<tr>
<td>OpenGL</td>
<td>DMA-Buf (soon)</td>
</tr>
<tr>
<td>Image</td>
<td></td>
</tr>
<tr>
<td>Alpha Hole</td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>Wayland</td>
</tr>
<tr>
<td>--------------------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>Xv</td>
<td>Buffers (fourcc)</td>
</tr>
<tr>
<td>OpenGL</td>
<td>DMA-Buf (soon)</td>
</tr>
<tr>
<td>Image</td>
<td></td>
</tr>
<tr>
<td>Alpha Hole</td>
<td>Zero copy!</td>
</tr>
<tr>
<td>X</td>
<td>Wayland</td>
</tr>
<tr>
<td>---</td>
<td>---------</td>
</tr>
<tr>
<td>Window reparenting</td>
<td>Subsurfaces</td>
</tr>
</tbody>
</table>
Under the Hood: Nesting Structures

X

Wayland

Window reparenting
Parent moves are atomic

Subsurfaces
Under the Hood: Nesting Structures

X

Window reparenting
Parent moves are atomic
Moving children isn’t

Wayland

Subsurfaces
Double buffered state
Atomic via parent commit
## Under the Hood: Nesting Structures

<table>
<thead>
<tr>
<th>X</th>
<th>Wayland</th>
</tr>
</thead>
<tbody>
<tr>
<td>Window reparenting</td>
<td>Subsurfaces</td>
</tr>
<tr>
<td>Parent moves are atomic</td>
<td>Double buffered state</td>
</tr>
<tr>
<td>Moving children isn’t</td>
<td>Atomic via parent commit</td>
</tr>
</tbody>
</table>

Scroll a browser window with lots of video elements...
Benefits to Tizen:

Coherent motion of complicated layouts
Again, “Every frame is perfect”
Under the Hood: Wayland security model

Applications have:

- No control over own window position
- No access to other client’s buffers
- No unfiltered access to input
Applications have:

No control over own window position
  No focus stealing

No access to other client’s buffers
  No color picker

No unfiltered access to input
  No xeyes
Under the Hood: Wayland security model

Benefits to Tizen:

Better protection from malicious software
Under the Hood: Wayland protocol extensions

Easy to add private extensions
Under the Hood: Wayland protocol extensions

Easy to add private extensions
Official protocol developed as Weston extensions
Under the Hood: Wayland protocol extensions

Easy to add private extensions
Gaussian blur?
Window orientation hints?
Under the Hood: Wayland protocol extensions

Benefits to Tizen:

Eye candy
Rapid innovation
Under the Hood: Portability

KMS works?
Weston works.
KMS works?
Weston works.

Don’t have to write Xv driver!
Under the Hood: Portability

KMS works? Weston works. *

*(GL has some additional requirements)*
Benefits to Tizen:

Quick bring-up on new platforms
Easy adoption

See Mauro Chehab’s talk “Bringing Tizen to a Raspberry PI 2 Near You!”
That's great – How do we use it?

Wayland and X share approximately 0% common API
That's great – How do we use it?

Wayland and X share approximately 0% common API

Rewrite all your applications?
That's great – How do we use it?

Wayland and X share approximately 0% common API

Rewrite all your applications?

Just kidding! Use EFL.
Enlightenment Foundation Libraries (EFL)

Data structures (lists, hash tables, etc)
Main loop event, I/O, timing core
Event queue and call handling
Canvas scene graph and rendering
Widgets/controls (buttons, sliders, etc)
Input method framework
Video & audio codec playback
IPC, D-Bus, network connectivity
Startup time

“time elementary_test”:
  Real: 0m1.645s
  User: 0m0.683s
  System: 0m0.017s

Memory Usage: 29.8 Mb
EFL Applications on X (cont)

Elementary Test
Peak memory: 9.1 MB
EFL Applications on Wayland

Startup time

“time elementary_test”:
Real: 0m1.227s
User: 0m0.420s
System: 0m0.003s

Memory Usage: 18.2 Mb
EFL Applications on Wayland (cont)

Elementary Test
Peak memory: 8.1 MB
Utilizing Wayland:

Decreased Startup Time:
  0.418s average per application

Decreased Memory Usage:
  11.6 Mb average per application

Similar functionality
Enlightenment: X vs Wayland

Startup time not measured

Enlightenment Memory Usage:
  In X11: 103.5 Mb
    X Server: 62 Mb
    Window Manager: 41.5 Mb
  Wayland: 54.9 Mb

Average Memory Savings: 48.6 Mb

Similar functionality