Tizen 3.0 's Window System Integration Layer of OpenGLES/EGL & Vulkan Driver

(libtpl-egl, vulkan-wsi-tizen)

Mun Gwan-gyeong
Software R&D Center
Samsung Electronics

Agenda

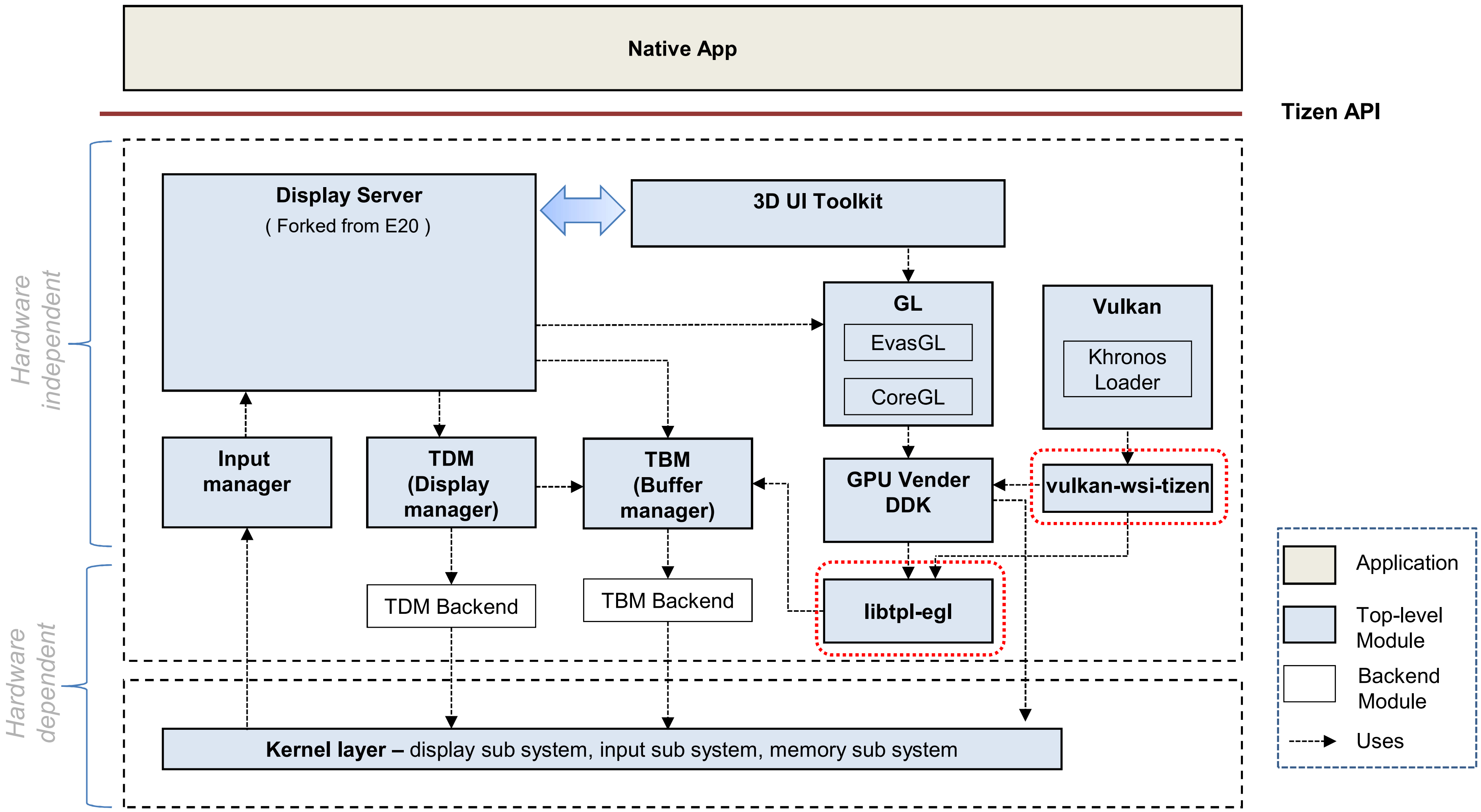
- Tizen 3.0 Window System Architecture

- Tizen 3.0 Window System Integration Layer of OpenGLES/EGL
  - libtpl-egl (Tizen Porting Layer for EGL)

- Tizen 3.0 Vulkan WSI for Tizen
  - vulkan-wsi-tizen

Tizen 3.0 Window System Architecture

Tizen 3.0 Window System Architecture



Components description

- **TPL-EGL** is an abstraction layer for surface and buffer management on Tizen platform aimed to implement the EGL porting layer of OpenGLES driver over various display protocols.

- **Vulkan-WSI-Tizen** wrapes vendor's vulkan ICDs and provides the WSI(Window-System Interface) for the tizen.

- **Tizen Buffer Manager (TBM)** provides the abstraction interface for the graphic buffer manager in Tizen.

- **Tizen Display Manager (TDM)** provides the abstraction interface for the display server, such a wayland server, to allow the direct access to graphics hardware in a safe and efficient manner as a display HAL.

Components description (cont.)

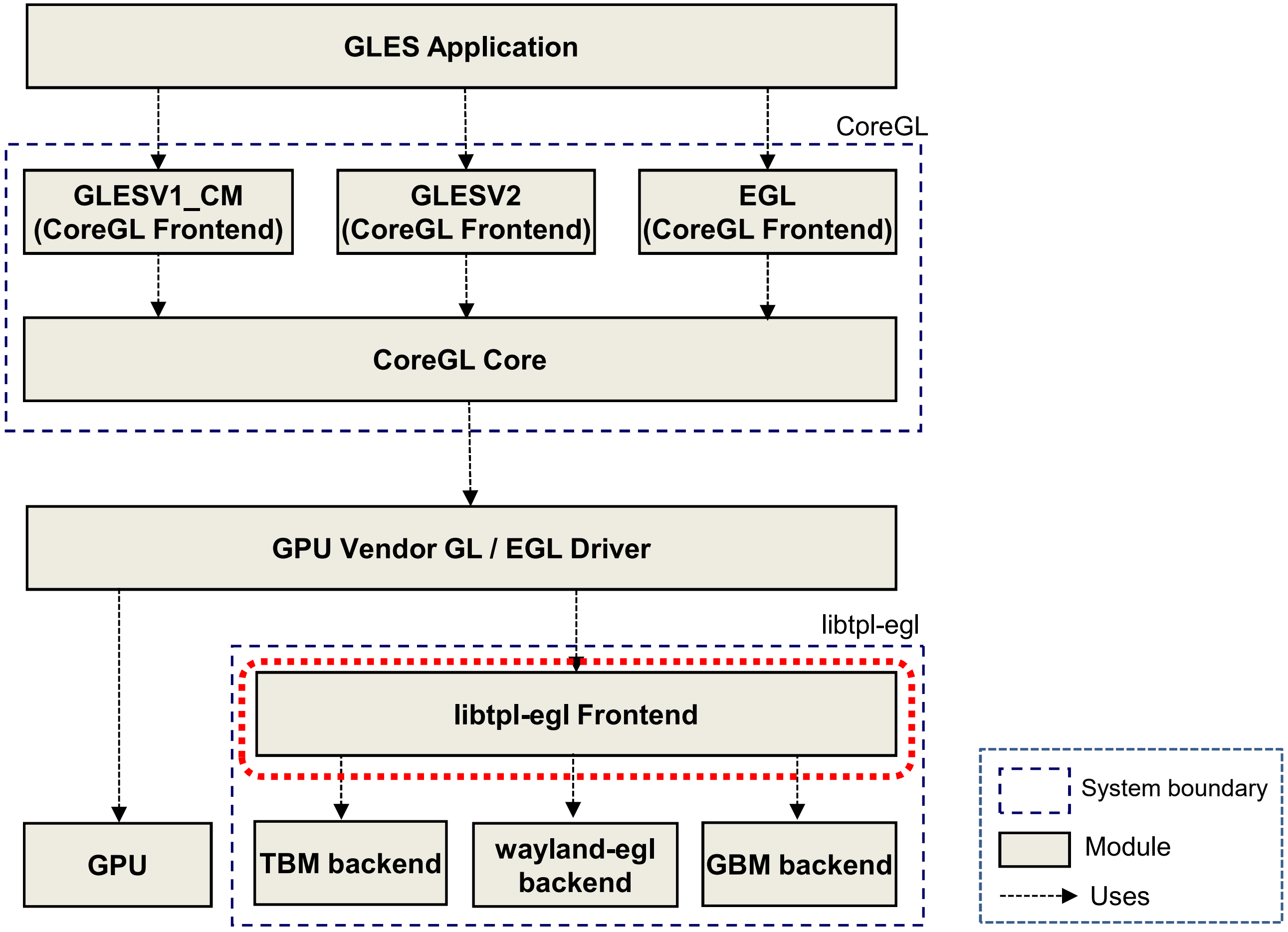
- **EvasGL** is a kind of Evas Object image for opengl and it is a GLES Wrapper.

- **CoreGL** is an injection layer of OpenGL ES that provides the following capabilities:

  - Support for driver-independent optimization (FastPath)
  - EGL/OpenGL ES debugging
  - Performance logging

Tizen Porting Layer for EGL (libtpl-egl)

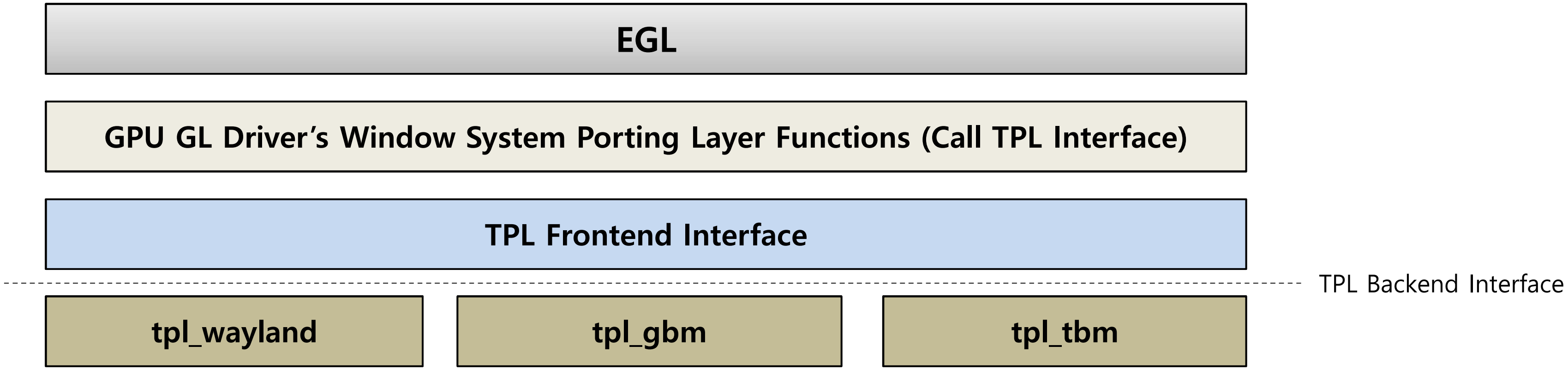
Tizen OpenGL ES and EGL Architecture



Tizen Porting Layer for EGL

Tizen Porting Layer(TPL) Architecture

- TPL provides implementation of EGL platform functions on Tizen platform


EGL
GPU GL Driver’s Window System Porting Layer Functions (Call TPL Interface)
TPL Frontend Interface
TPL Backend Interface
tpl_wayland
tpl_gbm
tpl_tbm

TPL?

- Background
  - Various window system protocols in Tizen
    - Wayland , gbm , tbm , ~~X11~~ (Tizen 3.0 Alpha)
  - Needs to separating common layer (frontend, duplicated code) and backend for maintaining
- Why TPL?
  - TPL-EGL APIs prevents burdens of EGL porting on various window system protocols.
  - Vendor GL Driver’s Window System Porting Layer functions treat only TPL-EGL APs.
  - If libtpl-egl has improved performance, then Vendor driver can get it without modification of code.

TPL Frontend Interface

- **Tizen Porting Layer Core Object**
  - **TPL Object**

    Base class for all TPL objects
  - **TPL Display**

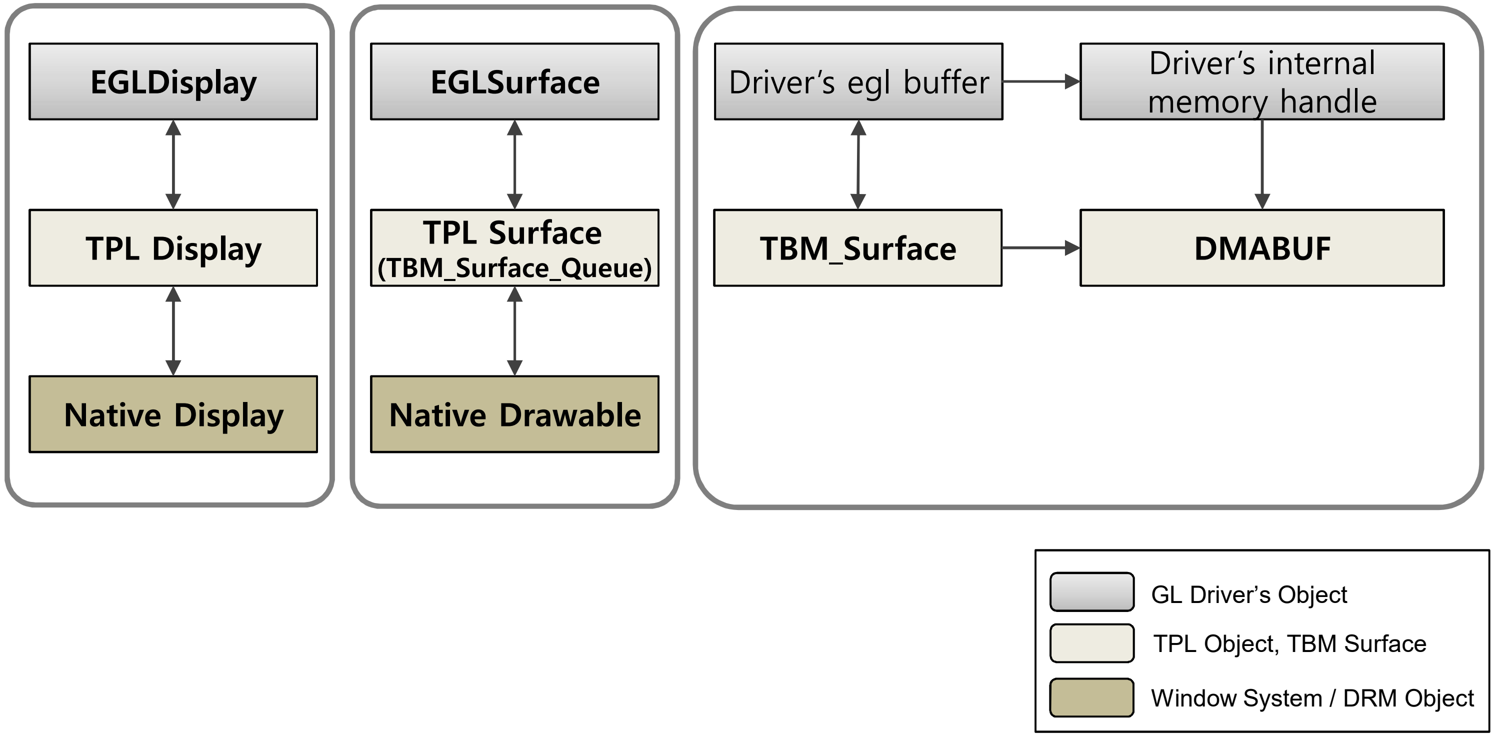
    Encapsulate native display object (wl_display, gbm_device, tbm_bufmgr)
  - **TPL Surface**

    Encapsulate native drawable object (wl_surface, gbm_surface, tbm_surface_queue_h)

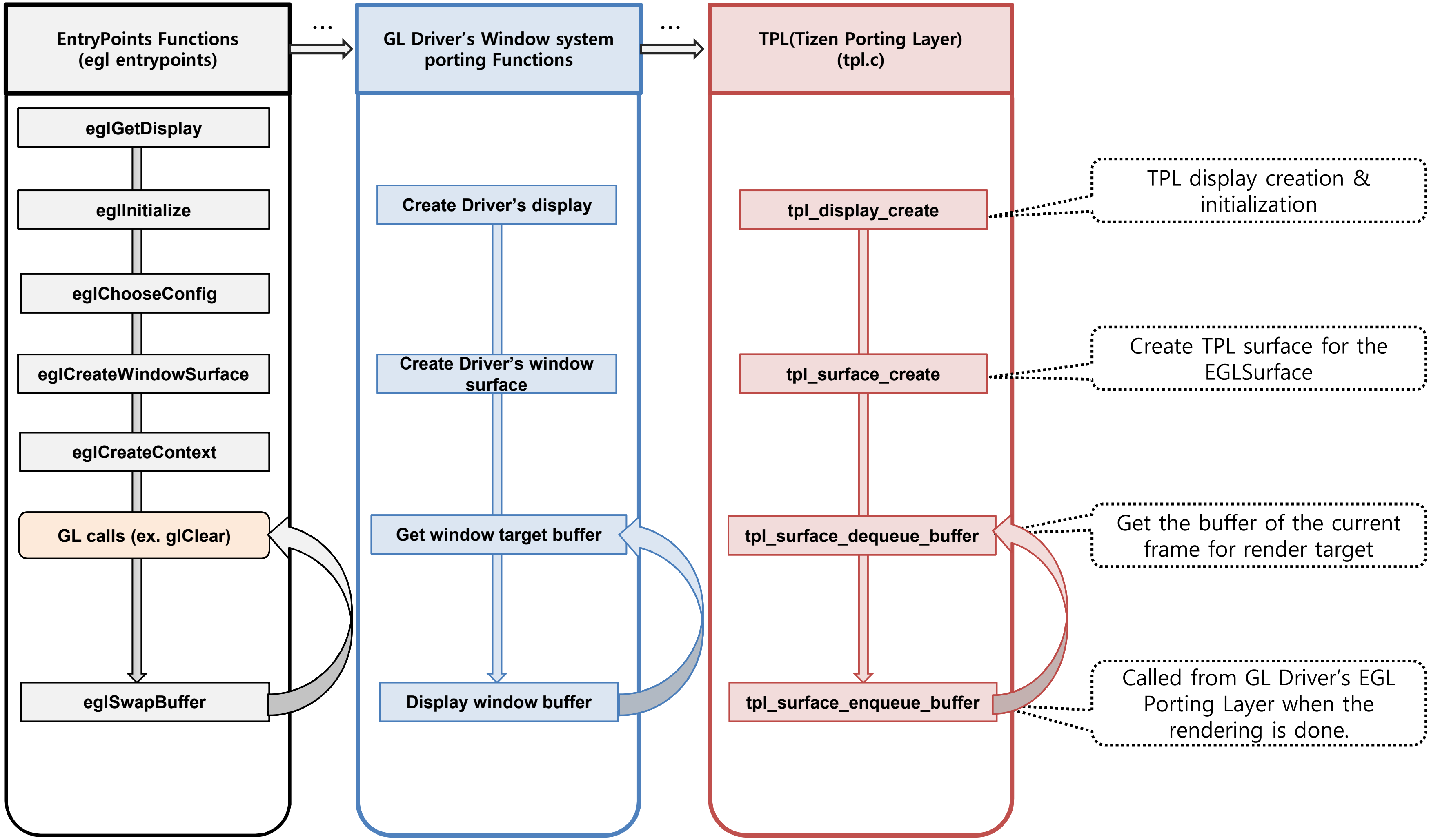
Tizen Porting Layer for EGL

TPL

- Provides TPL objects which correspond to EGL objects



GLES Drawing API Flow



TPL display creation & initialization

Create TPL surface for the EGLSurface

Get the buffer of the current frame for render target

Called from GL Driver's EGL Porting Layer when the rendering is done.

Simple example of the Tizen Porting Layer

```
tpl_display_t *dpy = tpl_display_create(...);
tpl_surface_t *sfc = tpl_surface_create(dpy, ...);
tbm_surface_h buf;
while (1)
{
    buf = tpl_surface_dequeue_buffer(sfc); // get buffer

    /* Draw something */

    tpl_surface_enqueue_buffer(sfc, buf); // post buffer
}
```

[pseudo code] Using libtpl-egl api

In the GPU vendor driver, the "Draw something" part is what the GPU frame builder does.
TPL-EGL exposes the native platform buffer as tbm_surface. If tbm backend uses drm_backend , GL Driver can get dma_buf from tbm_surface's buffer object.

TPL Frontend API (tpl_object)

TPL Object

- Base class for all TPL objects
- Provide common functionalities of all TPL objects

API	Description
tpl_object_reference	Increase reference count of the given TPL object
tpl_object_unreference	Decrease reference count and destroy it if it becomes 0
tpl_object_get_reference	Get reference count of the given TPL object
tpl_object_get_type	Get type of the object (display or surface)
tpl_object_set_user_data	Set user data and free callback for destruction
tpl_object_get_user_data	Get user data

TPL Frontend API (tpl_display)

TPL Display

- Encapsulate native display object (wl_display, gbm_device, tbm_bufmgr)
- Any other objects are created from TPL Display , they are inherited backend type from TPL Display.

API	Description
tpl_display_create	Creates the TPL-EGL display object for the given native display
tpl_display_get	Retrieve the TPL-EGL display for the given native display handle
tpl_display_get_native_handle	Get native handle of the given display
tpl_display_query_config	Query pixel format information
tpl_display_get_native_window_info	Query information on the given native window.
tpl_display_get_native_pixmap_info	Query information on the given native pixmap.
tpl_display_get_buffer_from_native_pixmap	Get native buffer from the given native pixmap.

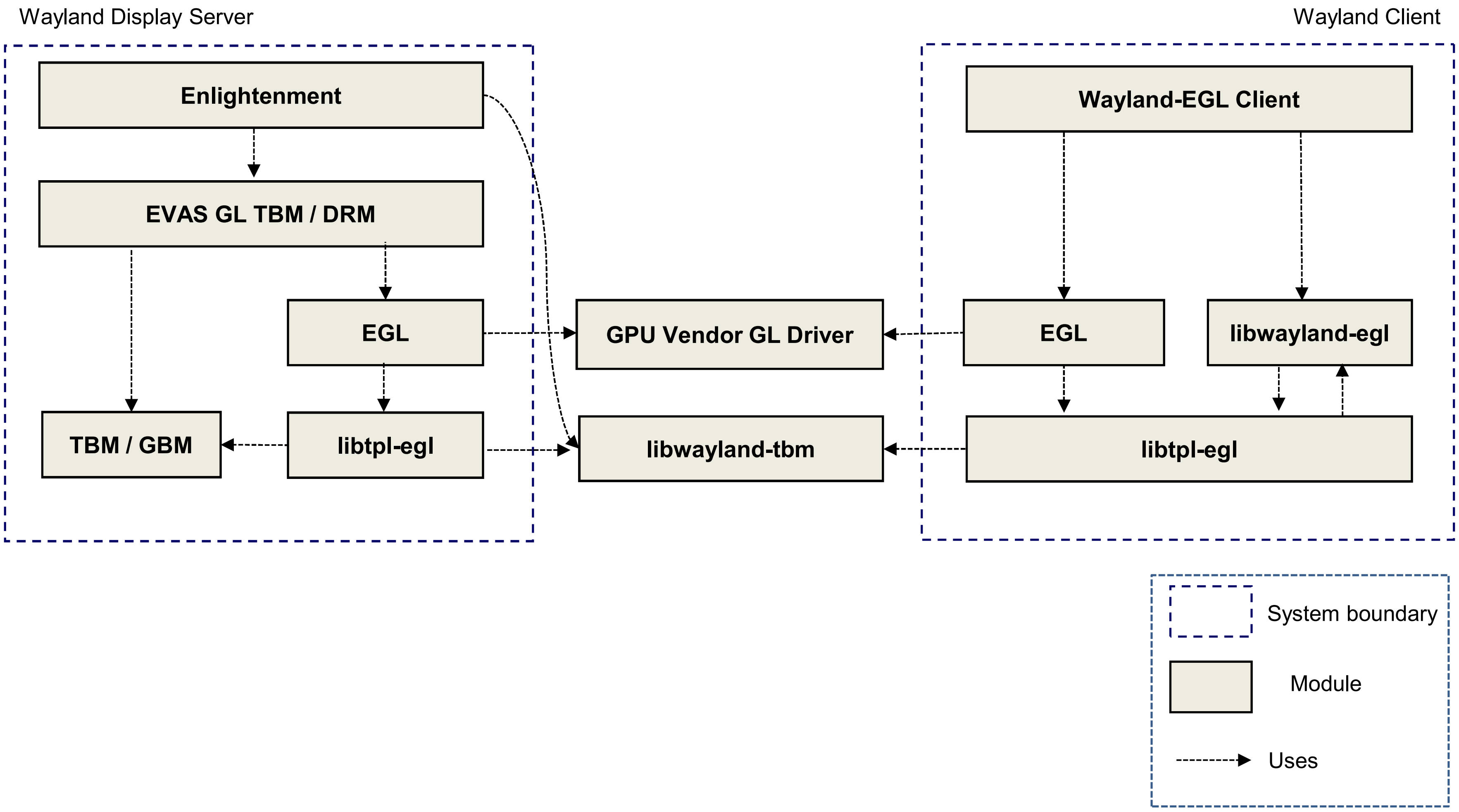
TPL Frontend API (tpl_surface)

TPL Surface

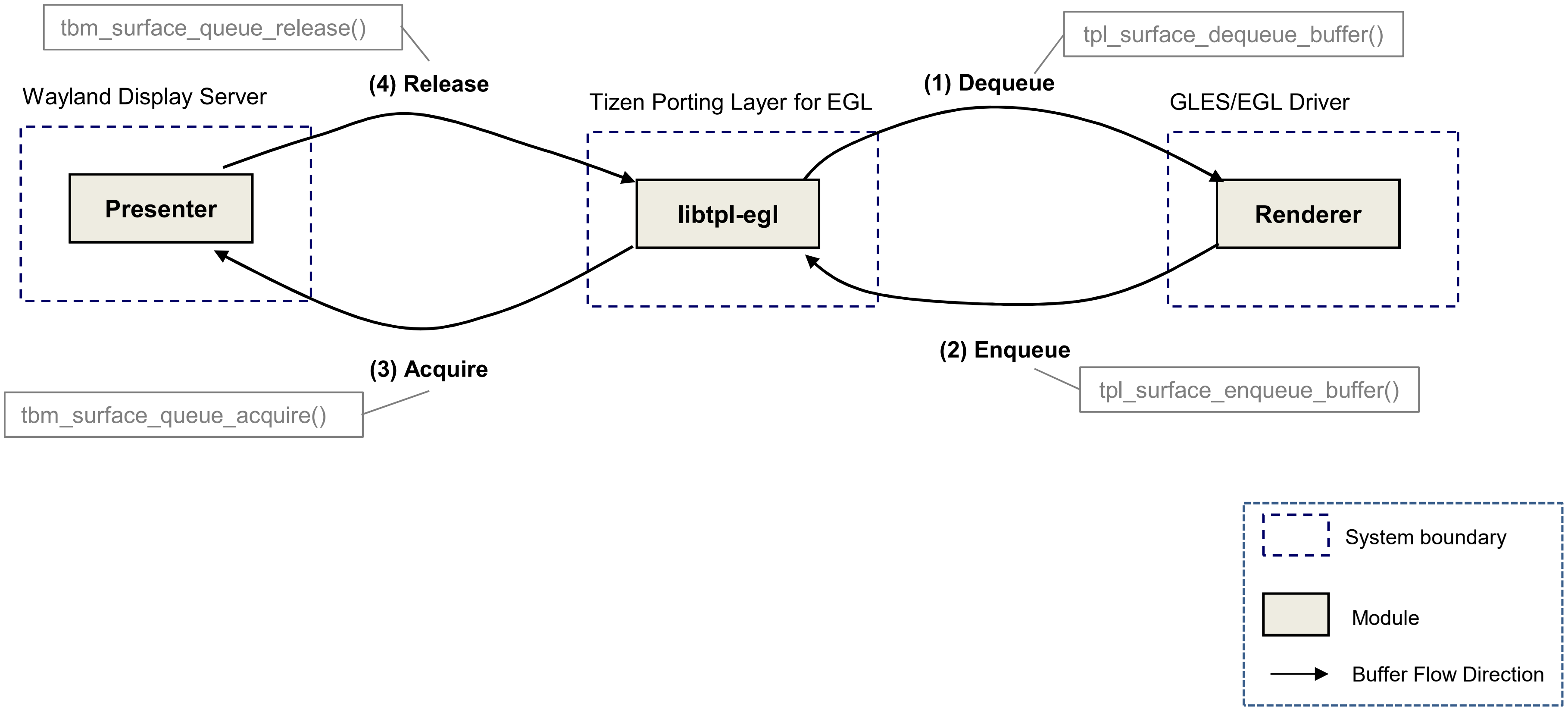
- Encapsulate native drawable object (wl_surface, gbm_surface, tbm_surface_queue_h)
- Main Features
  - Get the buffer for a surface
  - Post the buffer to a surface of screen

API	Description
tpl_surface_create	Create a TPL-EGL surface for the given native drawable
tpl_surface_get_display	Get TPL-EGL display of the given surface
tpl_surface_get_native_handle	Get native handle of the given surface
tpl_surface_get_type	Get type of the given surface (Window or Pixmap)
tpl_surface_get_size	Get size of the given surface
tpl_surface_dequeue_buffer	Get buffer (as TBM_SURFACE) of the current frame for the given surface
tpl_surface_validate	Check current buffer is valid
tpl_surface_set_post_interval	Set post interval
tpl_surface_get_post_interval	Get post interval
tpl_surface_enqueue_buffer	Post to screen

Wayland Server / Client on libtpl-egl



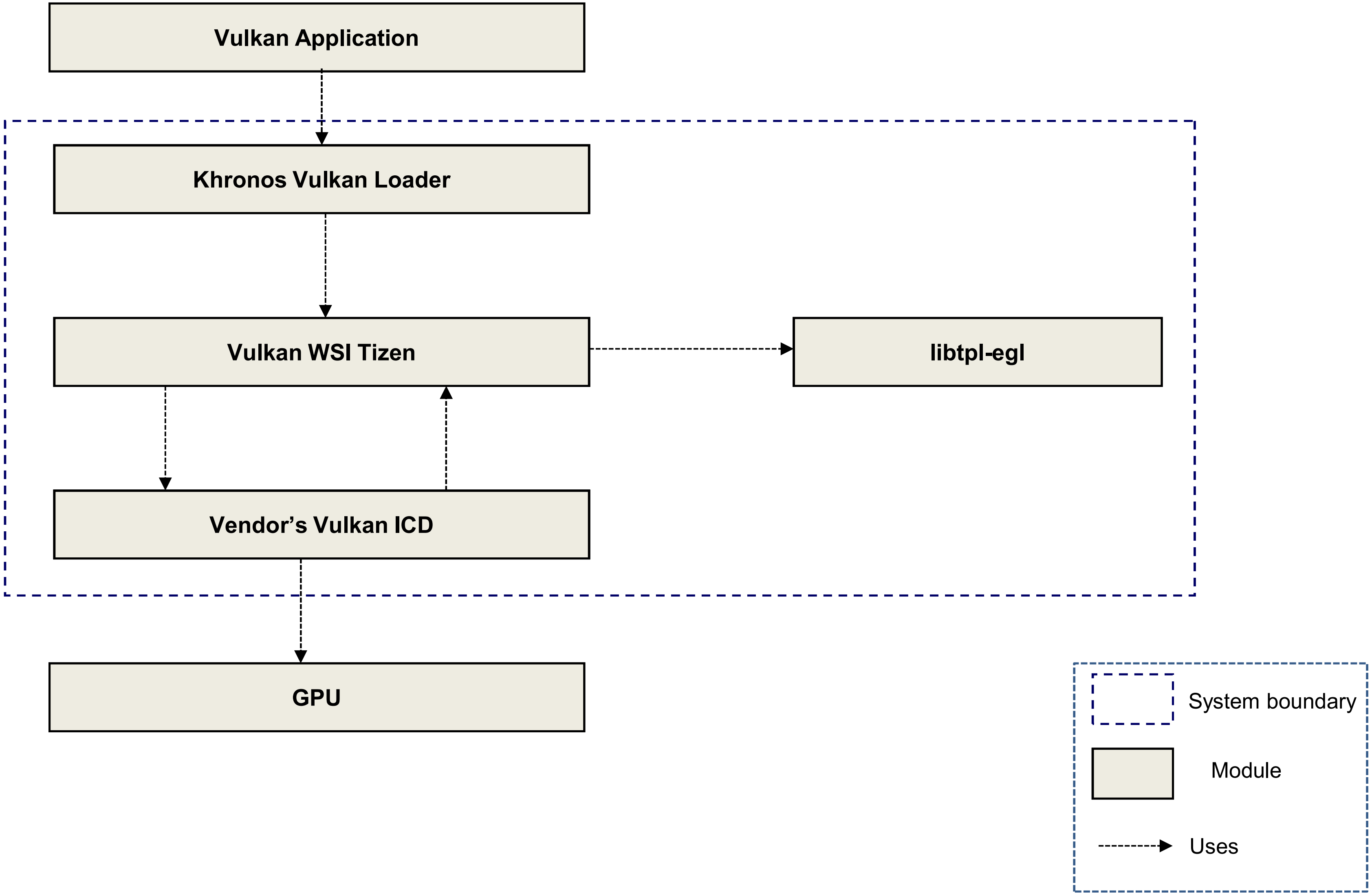
Buffer Flow (Wayland Server ↔ GLES/EGL Driver)



Buffer Flow Between the Wayland Server and GLES/EGL Driver

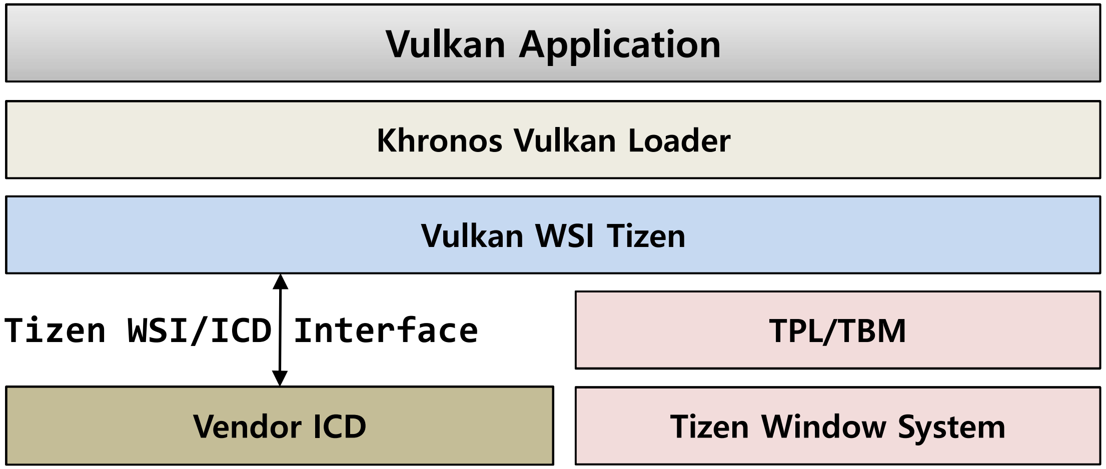
Vulkan WSI for Tizen (vulkan-wsi-tizen)

Tizen Vulkan Architecture



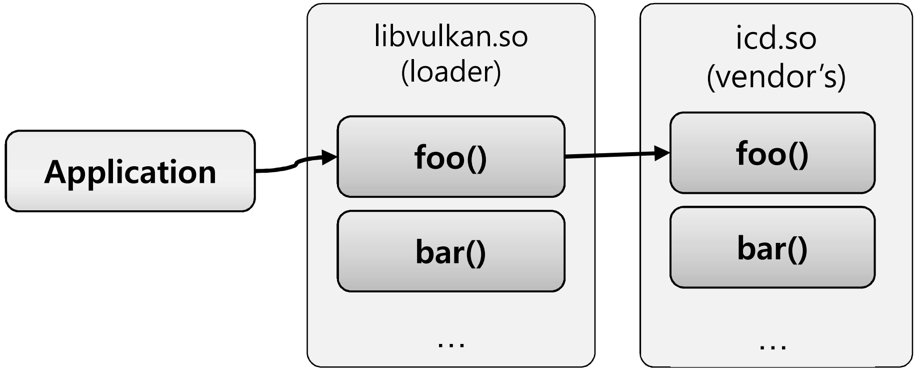
Vulkan WSI for Tizen

- **Objectives**
  - Applications should be able to use khronos vulkan loader
  - Do not modify khronos vulkan loader
  - Separate WSI binary across multiple vendor ICDs
  - Don’t do any platform specific things, use TPL instead
- **Architecture**
  - WSI wraps the ICD and act like a complete ICD



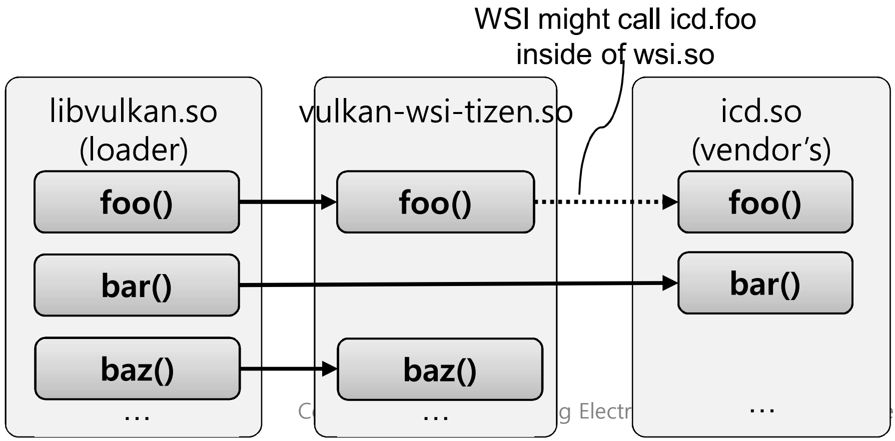
Vulkan Loader (Khronos Vulkan Loader)

- Loader exposes vulkan symbols to applications (libvulkan.so)
- Loader opens an ICD shared object file and dispatches ICD functions via vk_icdGetInstanceProcAddr()
  - This is recommended way according to the khronos loader document
- Application calls a loader function, then loader function finally calls the dispatched ICD function
  - Vulkan is layered architecture



Wrapping ICD (vulkan-wsi-tizen)

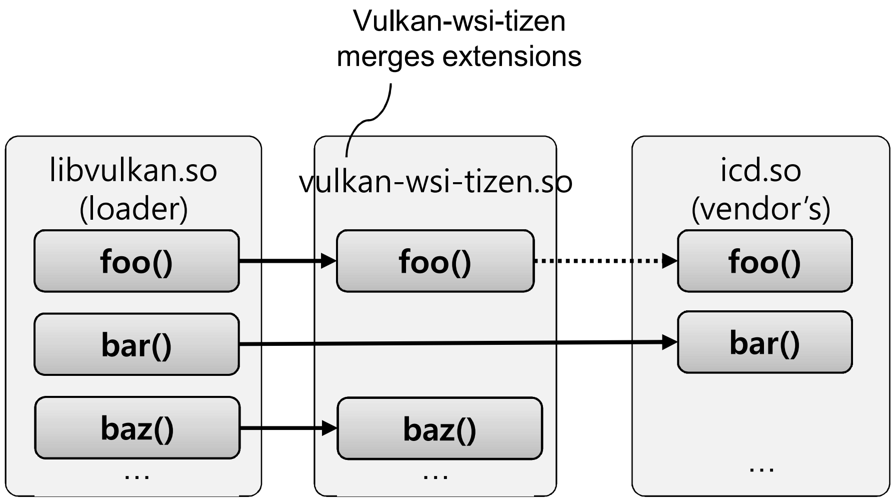
- Vulkan WSI Tizen acts like a complete ICD
  - Exposes vk_icdGetInstanceProcAddr() which dispatches all required vulkan functions
- Some functions are implemented in vulkan-wsi-tizen, while others come from vendor ICD
- API Hooks
  - Vulkan WSI Tizen hooks desired vulkan functions
  - Hooked vulkan-wsi-tizen functions are dispatched instead of ICD functions
  - vkGetInstanceProcAddr(), vkGetDeviceProcAddr() are hooked by default
    - If not, an (Vendor's) ICD function might be dispatched even though it is hooked by WSI



WSI might call icd.foo inside of wsi.so

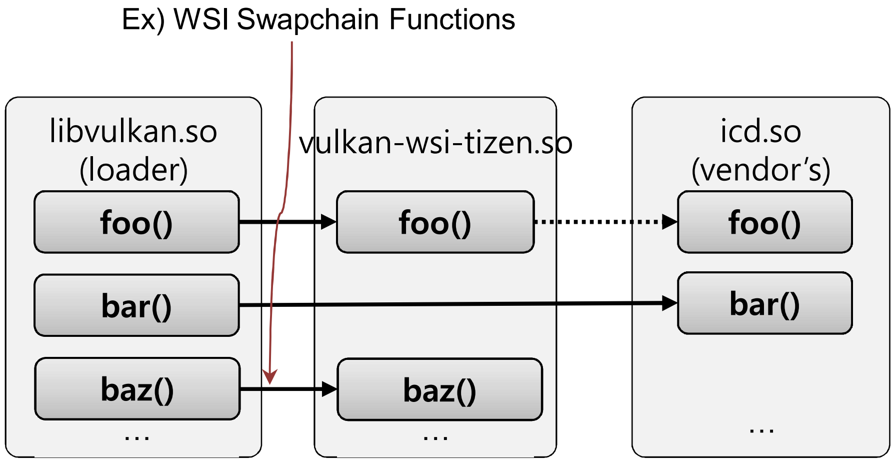
Extension Merge (vulkan-wsi-tizen)

- Extension Merge
  - vulkan-wsi-tizen merges extensions from Vendor ICD and vulkan-wsi-tizen's own extension
  - vulkan-wsi-tizen hooks extension enumeration functions
  - vkEnumerateInstanceExtensionProperties() in vulkan-wsi-tizen
    - Vendor ICD instance extension + VK_KHR_surface + VK_KHR_wayland_surface
  - vkEnumerateDeviceExtensionProperties() in vulkan-wsi-tizen
    - Vendor ICD device extension + VK_KHR_swapchain



WSI Surface Functions (Khronos Vulkan Loader)

- WSI Surface Functions
  - Surface functions are implemented in the khronos loader
  - Surface object is passed to the vulkan-wsi-tizen when other WSI function is called
    - ex) vkCreateSwapchainKHR
  - Data structure for the loader surface object can be accessed via vk_icd.h (Khronos Vulkan Loader's header file)

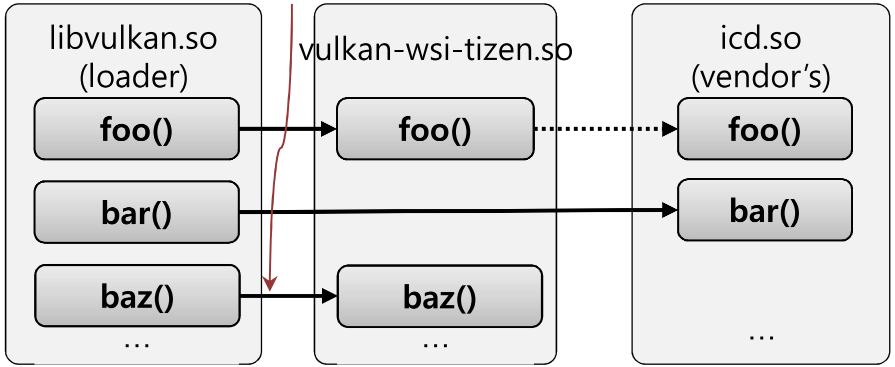


Copyright © 2016 Samsung Electronics, Co., Ltd. All rights reserved.

WSI Functions (vulkan-wsi-tizen)

- WSI functions except surface functions are implemented and hooked
- WSI function categories
  - Surface capability query functions
    - Formats, presentation support, …
    - ex) vkGetPhysicalDeviceSurfaceCapabilitiesKHR(), vkGetPhysicalDeviceSurfaceFormatsKHR() …
  - Swapchain functions
    - ex) vkCreateSwapchainKHR(), vkGetSwapchainImagesKHR(), vkAcquireNextImageKHR(), vkQueuePresentKHR() …
  - Display functions
    - Required when presenting directly to a display device

Ex) WSI Swapchain Functions

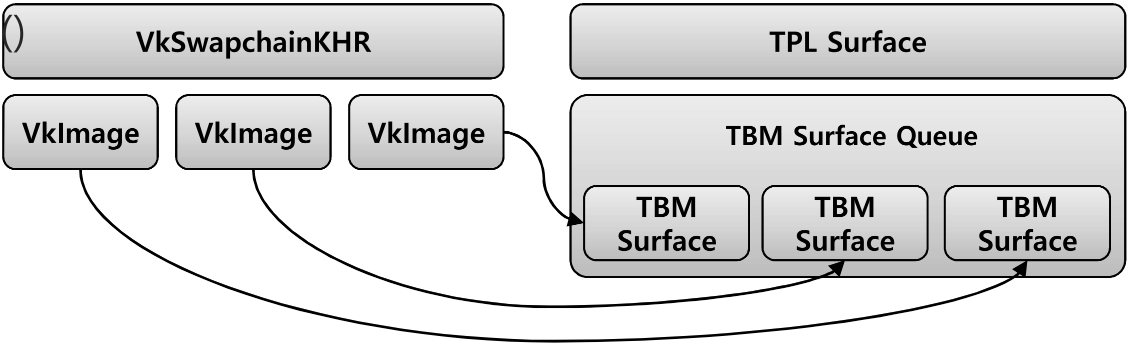


Swapchain related API (vulkan-wsi-tizen)

- Swapchain
  - Manages image (buffer) queue
  - vkAcquireNextImageKHR()
    - Acquire a next image from the presentation engine
  - vkQueuePresentKHR()
    - Present the given image to the presentation engine
  - Implemented using TPL surface

- vkAcquireNextImageKHR()
  - tpl_surface_dequeue_buffer()
  - Find index of the dequeued buffer and return

- vkQueuePresentKHR()
  - tpl_surface_enqueue_buffer()



Vulkan WSI Tizen ↔ Vendor's ICD interface

- Vulkan WSI Tizen ↔ Vendor's ICD interface
  - Vendor's ICD should provide functions required by the Vulkan WSI Tizen
  - vk_tizen.h
    - Defines functions ICD should provides
    - Vulkan WSI Tizen should be able to dispatch those functions via Vendor ICD's vk_icdGetInstanceProcAddr()

- vkCreateImageFromNativeBufferTIZEN()
  - It creates a VkImage from a tizen native buffer (tbm_surface_h)
  - It is called by vkCreateSwapchainKHR() of vulkan-wsi-tizen
  - Arguments
    - [in] VkDevice
      - VkDevice is passed by vkCreateSwapchainKHR()
    - [in] tbm_surface_h
      - Native tizen buffer
    - [in] const VkImageCreateInfo *
      - Properties of the native tizen buffer (dimension, format, …)
    - [in] const VkAllocationCallbacks *
      - Allocation callbacks used for host memory allocation
    - [out] VkImage *
      - Vendor ICD should create vkImage from tbm_surface.
      - vkAcquireNextImageKHR() uses this VkImage.

Vulkan WSI Tizen ↔ Vendor's ICD (cont.)

- vkQueueSignalReleaseImageTIZEN()
  - When the vendor's vulkan driver ends up the handling of vkImage and it is ready to present (all waiting semaphores are triggered), Vendor ICD notifies to vulkan-wsi-tizen (NativeFenceFd is created by Vendor Driver.)
  - It is called by vkQueuePresentKHR() of vulkan-wsi-tizen
  - Arguments
    - [in] VkQueue
      - VKQueue is passed by vkQueuePresentKHR()
    - [in] uint32_t
      - waitSemaphoreCount is passed by VkPresentInfoKHR of vkQueuePresentKHR()
    - [in] const VkSemaphore *
      - WaitSemaphore list is passed by VkPresentInfoKHR of vkQueuePresentKHR()
    - [in] VkImage
      - VkImage index is passed by VkPresentInfoKHR of vkQueuePresentKHR()
    - [out] int *NativeFenceFd
      - Vendor ICD should create NativeFenceFd from WaitSemaphore list.
      - vulkan-wsi-tizen waits NativeFeceFd by tbm_sync_fence_wait().

Vulkan WSI Tizen ↔ Vendor's ICD (cont.)

- vkAcquireImageTIZEN()
  - It notifies the acquired Image which is ready to use to the Vendor's Vulkan Driver.
  - It is called by vkAcquireNextImageKHR() of vulkan-wsi-tizen
  - Arguments
    - [in] VkDevice
      - VkDevice is passed by vkAcquireNextImageKHR()
    - [in] VkImage
      - VkImage index is passed by vkAcquireNextImageKHR()
    - [in] int nativeFenceFD
      - Vulkan driver should wait this nativeFenceFD until Display Server triggers it. (Display Server uses tbm_sync_timeline_inc() for triggering)
      - nativeFenceFD is created by tbm_sync_fence_create()
    - [in] VkSemaphore
      - Vendor ICD connects VkSemaphore to nativeFenceFD
      - When nativeFenceFD is triggered, Vendor ICD signals VkSemaphore
    - [in] VkFence
      - Vendor ICD connects VkFence to nativeFenceFD
      - When nativeFenceFD is triggered, Vendor ICD signals VkFence

Supported WSI Spec (Current State)

Surface & Swapchain Functions

Function	Status
vkCreateWaylandSurfaceKHR	Provided by khronos loader
vkDestroySurfaceKHR	Provided by khronos loader
vkGetPhysicalDeviceWaylandPresentationSupportKHR	Done
vkGetPhysicalDeviceSurfaceSupportKHR	Done
vkGetPhysicalDeviceSurfaceCapabilitiesKHR	Done
vkGetPhysicalDeviceSurfaceFormatsKHR	Done
vkGetPhysicalDeviceSurfacePresentModesKHR	Done
vkCreateSwapchainKHR	Done
vkCreateSharedSwapchainKHR	Not Implemented Yet
vkDestroySwapchainKHR	Done
vkGetSwapchainImagesKHR	Done
vkAcquireNextImageKHR	Done
vkQueuePresentKHR	Done

Supported WSI Spec

Present Modes

Modes	Status
VK_PRESENT_MODE_IMMEDIATE_KHR	Not Implemented Yet
VK_PRESENT_MODE_MAILBOX_KHR	Done
VK_PRESENT_MODE_FIFO_KHR	Not Implemented Yet
VK_PRESENT_MODE_FIFO_RELAXED_KHR	Not Implemented Yet

Display Functions

Function	Status
vkCreateDisplaySurfaceKHR	Provided by khronos loader
vkGetPhysicalDeviceDisplayPropertiesKHR	Not Implemented Yet
vkGetPhysicalDeviceDisplayPlanePropertiesKHR	Not Implemented Yet
vkGetDisplayPlaneSupportedDisplaysKHR	Not Implemented Yet
vkGetDisplayModePropertiesKHR	Not Implemented Yet
vkCreateDisplayModeKHR	Not Implemented Yet
vkGetDisplayPlaneCapabilitiesKHR	Not Implemented Yet

References

Project Git repogitory (https://review.tizen.org/gerrit/#/admin/projects/)

Project	Repository	Description
libtpl-egl	platform/core/uifw/libtpl-egl	Tizen Porting Layer for EGL
vulkan-wsi-tizen	platform/core/uifw/vulkan-wsi-tizen	vulkan wsi tizen icd, it wrapps vendor icd and provides wsi for tizen
libtbm	platform/core/uifw/libtbm	The library for the Tizen Buffer Manager
coregl	platform/core/uifw/coregl	An injection layer of OpenGL ES / EGL
wayland-tbm	platform/core/uifw/wayland-tbm	Wayland tbm is a protocol for graphics memor y management for Tizen
emulator-yagl	platform/adaptation/emulator/emulator-yagl	OpenGL ES / EGL driver for the emulator
tpl-novice	platform/core/uifw/ws-testcase	Novice test framework for TPL

libtpl-egl Reference Driver

- The Emulator YAGL (OpenGLES / EGL driver for the emulator) is implemented by libtpl-egl.
- The following commit explains how to port the driver with libtpl-egl from the traditional drm-based driver.
- Porting YAGL to the Tizen platform https://review.tizen.org/gerrit/#/c/67921/

Thank you