Swapchains Unchained!

(What you need to know about Vulkan WSI)

Alon Or-bach, Chair, Vulkan System Integration Sub-Group – May 2016

@alonorbach (disclaimers apply!)

Intro to Vulkan Window System Integration

- **Explicit control for acquisition and presentation of images**
  - Designed to fit the Vulkan API and today's compositing window systems
- **Not all extensions are supported by every platform**
  - You MUST check and enable the extensions your app/engine uses!!!
- **Today's presentation should help you get presentation working**
  - Learn how to present through a swapchain
  - Overview of Vulkan objects used by the WSI extensions

WSI Jargon Buster

- **Platform**
  Our terminology for an OS / window system e.g. Android, Windows, Wayland, X11 via XCB
- **Presentation Engine**
  The platform's compositor or display engine
- **Application**
  Your app or game engine

How many WSI extensions are there?

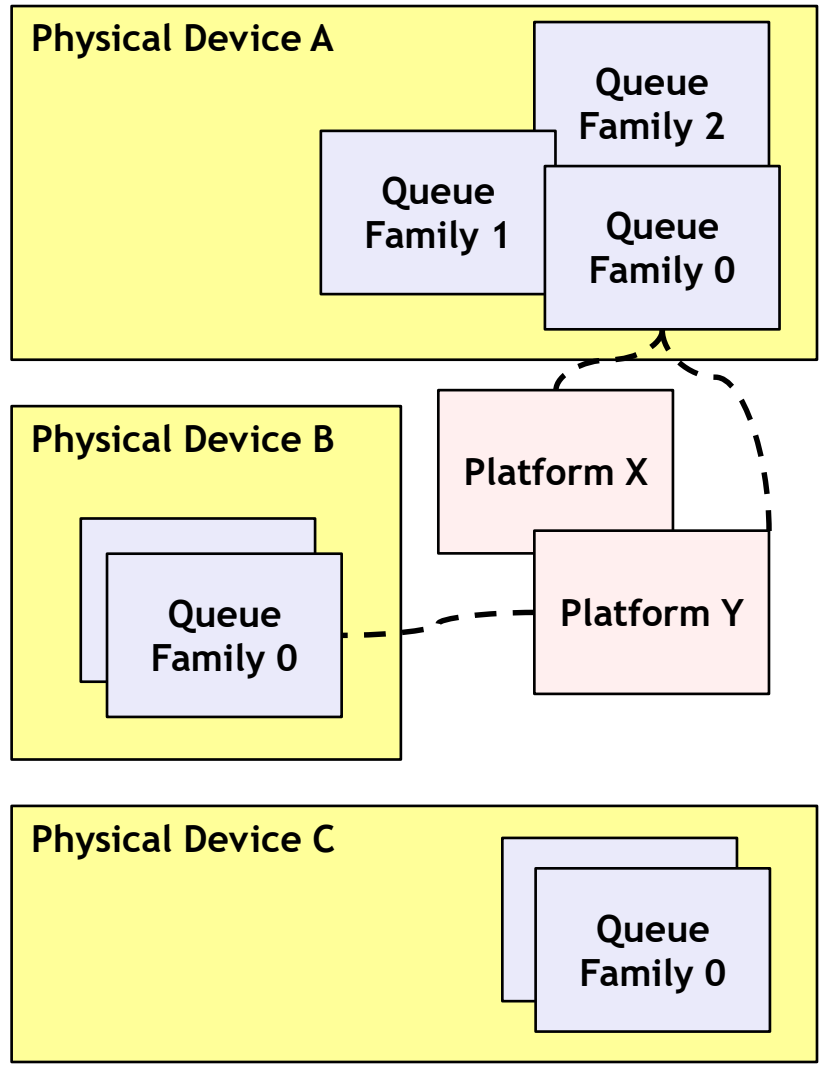
- **Two cross-platform instance extensions**
  - VK_KHR_surface
  - VK_KHR_display
- **Six (platform) instance extensions**
  - VK_KHR_android_surface
  - VK_KHR_mir_surface
  - VK_KHR_wayland_surface
  - VK_KHR_win32_surface
  - VK_KHR_xcb_surface
  - VK_KHR_xlib_surface
- **Two cross-platform device extensions**
  - VK_KHR_swapchain
  - VK_KHR_display_swapchain

Vulkan Surfaces

- **VkSurfaceKHR**
  - Vulkan's way to encapsulate a native window / surface

Unlike an EGLSurface, creating a Vulkan Surface doesn't mean you've got your render targets created …yet

- **Platform-independent surface queries**
  - Find out crucial information about your surface's properties
    - Such as format, transform, image usage
  - Some platforms provide additional queries
- **Presentation support is per queue family**
  - An implementation may support multiple platforms e.g. both xlib and xcb
  - Or may not support presentation at all



Vulkan Swapchains: VK_KHR_swapchain

- **Array of presentable images associated with a surface**
  - Application requests a minimum number of presentable images
  - Implementation creates at least that number
  - Implementation may have a limit
- **Upfront allocation of presentable images**
  - No allocation hitching at crucial moment
  - Pre-record fixed content command buffers
- **Present mode determines behavior**
  - FIFO support mandatory
  - Platforms can offer mailbox, immediate, FIFO relaxed

```
const VkSwapchainCreateInfoKHR createInfo =
{
 VK_STRUCTURE_TYPE_SWAPCHAIN_CREATE_INFO_KHR, // sType
 NULL, // pNext
 0, // flags
 mySurface, // surface
 desiredNumberOfPresentableImages, // minImageCount
 surfaceFormat, // imageFormat
 surfaceColorSpace, // imageColorSpace
 myExtent, // imageExtent
 1, // imageArrayLayers
 VK_IMAGE_USAGE_COLOR_ATTACHMENT_BIT, // imageUsage
 VK_SHARING_MODE_EXCLUSIVE, // imageSharingMode
 0, // queueFamilyIndexCount
 NULL, // pQueueFamilyIndices
 surfaceProperties.currentTransform, // preTransform
 VK_COMPOSITE_ALPHA_INHERIT_BIT_KHR, // compositeAlpha
 swapchainPresentMode, // presentMode
 VK_TRUE, // clipped
 VK_NULL_HANDLE // oldSwapchain
};
```

FIFO is like eglSwapInterval = 1
Mailbox/Immediate is like eglSwapInterval 0
FIFO relaxed is like EXT_swap_control_tear

Vulkan Swapchains: They're good!

- **Application knows which image within a swapchain it is presenting**
  - Content of image preserved between presents
- **Application is responsible for explicitly recreating swapchains - no surprises**
  - Platform informs app if current swapchain
    - Suboptimal: e.g. after window resize, swapchain still usable for present via image scaling
    - Surface Lost: swapchain no longer usable for present
  - Application is responsible to create a new swapchain

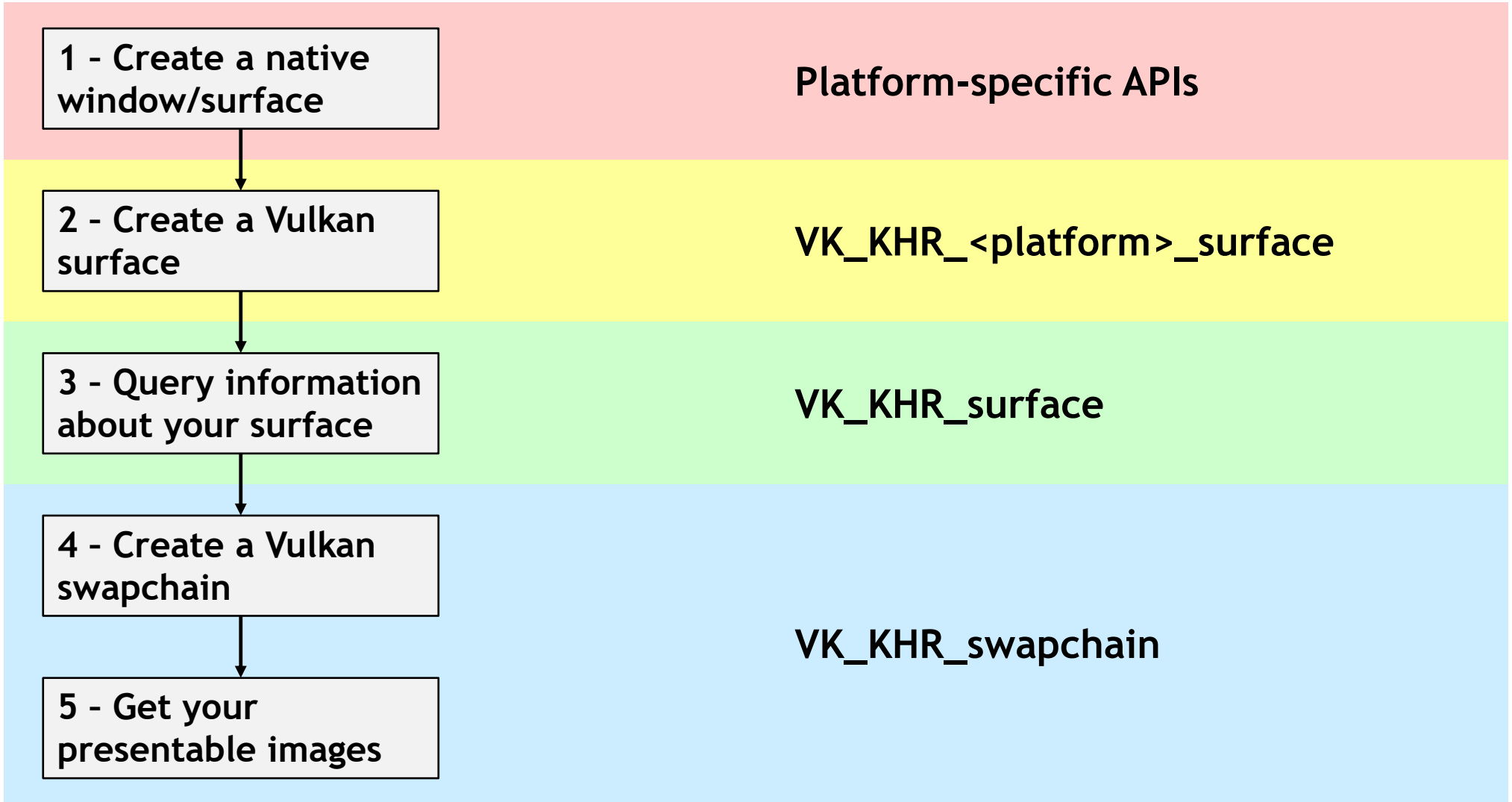
In EGL, the EGLSurface may be resized by the platform after an eglSwapBuffers call.
Vulkan requires the application to intervene

Vulkan Swapchains: They're jolly good!

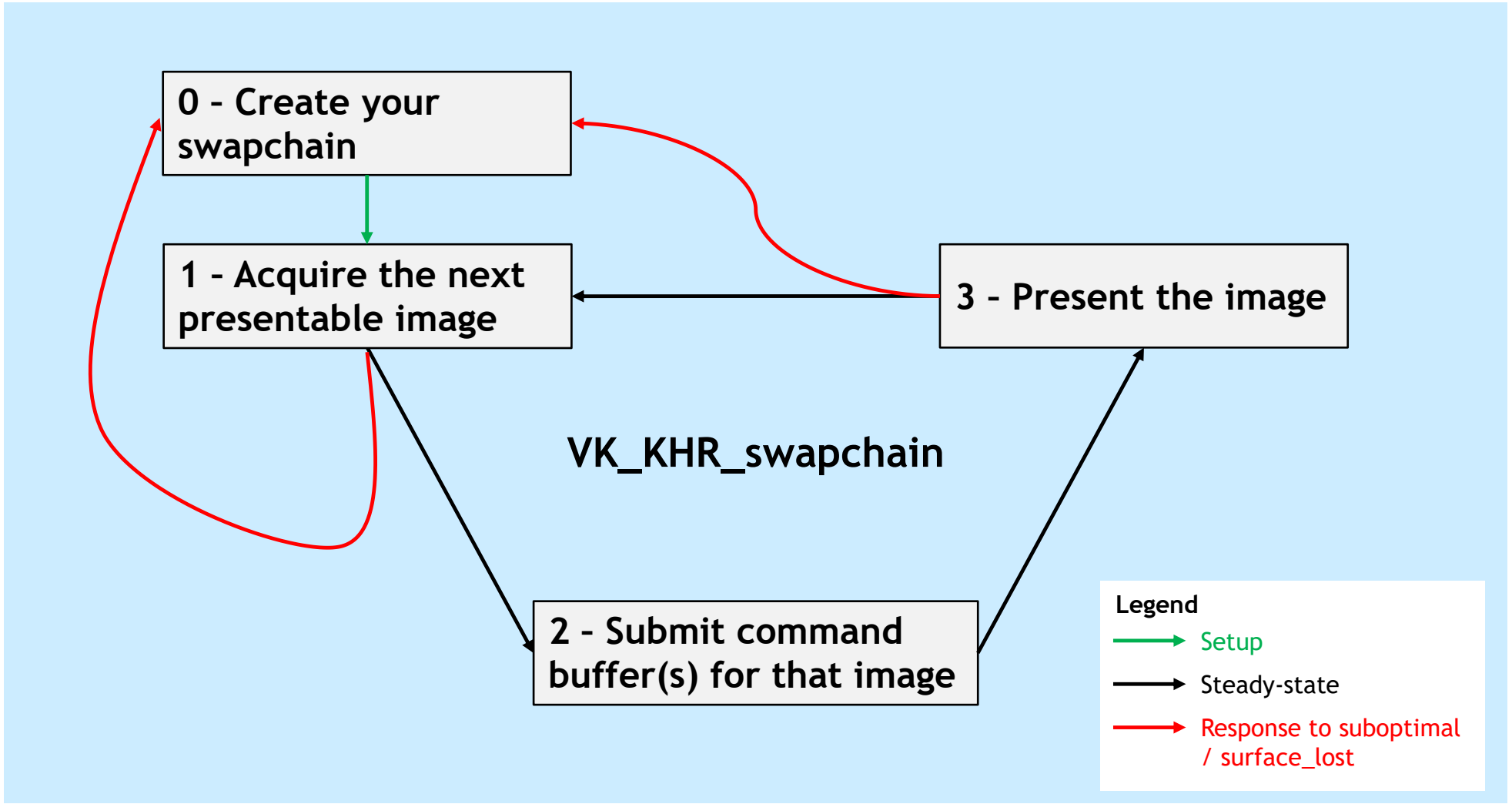
- **Presenting and acquiring are separate operations**
  - No need to submit a new image to acquire another one, unless presentation engine cannot release it
- **Application must only modify presentable images it has acquired**
- **Presentation engine must only display presentable images that have been presented!**

In EGL, calling eglSwapBuffers both presents the current back buffer **and** acquires a new one
Vulkan splits this up into separate operations

Steps to setup your presentable images

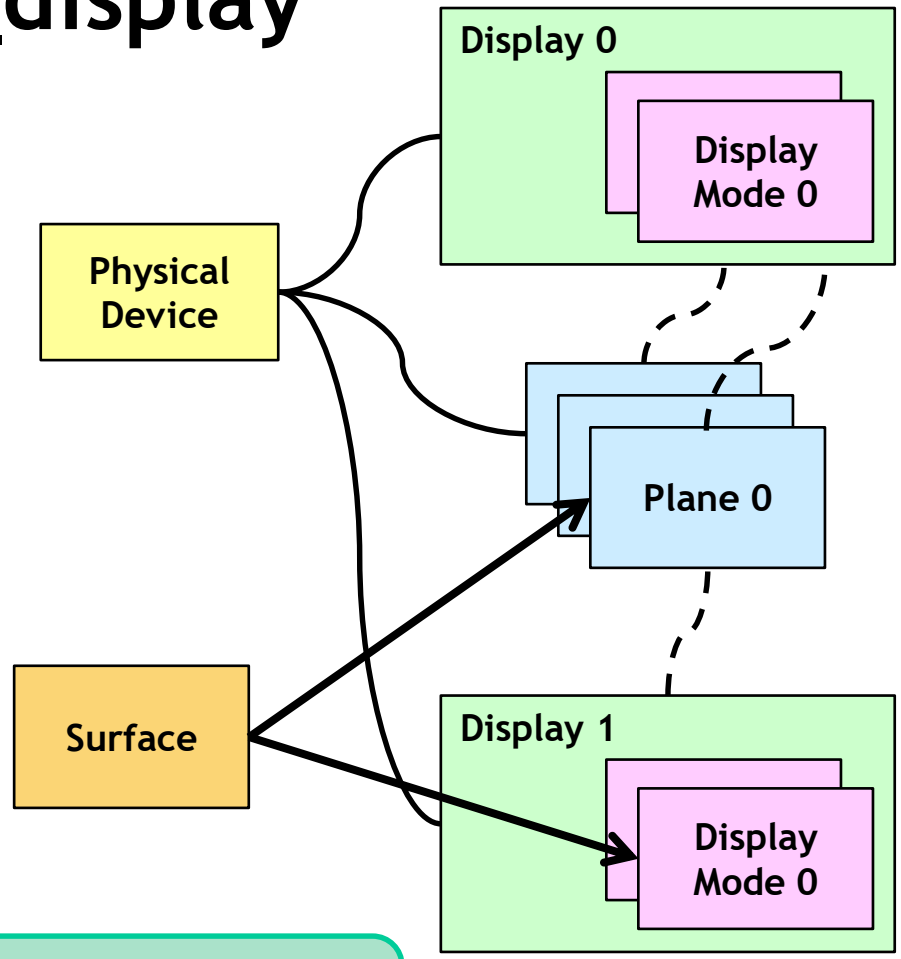


Vulkan Frame Loop - as easy as 1-2-3!



Vulkan Displays: VK_KHR_display

- **Vulkan's way to discover display devices (screens, panels) outside a window system**
  - Reminder: Not supported on all platforms
- **Defines VkDisplayKHR and VkDisplayModeKHR objects**
  - Represent the display devices and the modes they support connected to a VkPhysicalDevice
  - Determine if a display supports multiple planes that are blended together
- **Enables creation of a VkSurfaceKHR to represent a display plane**



A Vulkan display represents an actual display!
(Whereas an EGLDisplay is actually just a connection to a driver – like a Vulkan Device)

VK_KHR_display_swapchain

- **Extends the information provided at vkQueuePresentKHR**
  - What region to present from the swapchain image
  - What region to present to on the display
  - Whether the display should persist the image
- **Adds ability to create a shared swapchain**
  - Swapchain that takes multiple VkSwapchainCreateInfoKHR structs
  - Allows multiple displays to be presented to simultaneously
  - No guarantee that presents are atomic ...presently!

Any question?

alon.orbach@samsung.com
@alonorbach