The Swap Chain

How OpenGL Thinks of Framebuffers

How Vulkan Thinks of Framebuffers – the Swap Chain

What is a Swap Chain?

Vulkan does not use the idea of a “back buffer”. So, we need a place to render into before moving an image into place for viewing. This is called the Swap Chain.

In essence, the Swap Chain manages one or more image objects that form a sequence of images that can be drawn into and then given to the Surface to be presented to the user for viewing.

Swap Chains are arranged as a ring buffer

Swap Chains are tightly coupled to the window system.

After creating the Swap Chain in the first place, the process for using the Swap Chain is:

1. Ask the Swap Chain for an image
2. Render into it via the Command Buffer and a Queue
3. Return the image to the Swap Chain for presentation
4. Present the image to the viewer (copy to “front buffer”)
We Need to Find Out What our Display Capabilities Are

VulkanDebug.txt output for an Nvidia A6000:

** Init08Swapchain ****
kGetPhysicalDeviceSurfaceCapabilitiesKHR:
  minImageCount = 2 ; maxImageCount = 8
  currentExtent = 1024 x 1024
  minImageExtent = 1024 x 1024
  maxImageExtent = 1024 x 1024
  maxImageArrayLayers = 1
  supportedTransforms = 0x0001
  currentTransform = 0x0001
  supportedCompositeAlpha = 0x0001
  supportedUsageFlags = 0x009f

vkGetPhysicalDeviceSurfaceSupportKHR:
  ** This Surface is supported by the Graphics Queue **

Found 3 Surface Formats:
  0:        44                0  VK_COLOR_SPACE_SRGB_NONLINEAR_KHR
  1:        50                0  VK_COLOR_SPACE_SRGB_NONLINEAR_KHR
  2:        64                0  VK_COLOR_SPACE_SRGB_NONLINEAR_KHR

Found 4 Present Modes:
  0:        2   VK_PRESENT_MODE_FIFO_KHR
  1:        3   VK_PRESENT_MODE_FIFO_RELAXED_KHR
  2:        1   VK_PRESENT_MODE_MAILBOX_KHR
  3:        0   VK_PRESENT_MODE_IMMEDIATE_KHR

Here’s What the Vulkan Spec Has to Say About Present Modes, II

*Vk_present_mode_immediate_KHR* specifies that the presentation engine does not wait for a vertical blanking period to update the current image, meaning this mode may result in visible tearing. No internal queuing of presentation requests is needed, as the requests are applied immediately.

*Vk_present_mode_mailbox_KHR* specifies that the presentation engine waits for the next vertical blanking period to update the current image. Tearing cannot be observed. An internal single-entry queue is used to hold pending presentation requests. If the queue is full when a new presentation request is received, the new request replaces the existing entry, and any images associated with the prior entry become available for re-use by the application. One request is removed from the queue and processed during each vertical blanking period in which the queue is non-empty.

*Vk_present_mode_fifo_KHR* specifies that the presentation engine waits for the next vertical blanking period to update the current image. Tearing cannot be observed. An internal queue is used to hold pending presentation requests. New requests are appended to the end of the queue, and one request is removed from the beginning of the queue and processed during each vertical blanking period in which the queue is non-empty. This is the only valid value of present mode that is required to be supported.

*Vk_present_mode_continuousKHR* specifies that the presentation engine periodically updates the current image on its regular refresh cycle. The application is only required to make one initial presentation request, after which the presentation engine will automatically update the current image without any need for further presentation requests. The application can indicate the image contents have been updated by making a presentation request, but this does not guarantee the timing of when it will be updated. This mode may result in visible tearing if rendering to the image is not timed correctly.
Creating a Swap Chain

```
vkCreateSwapchainKHR
```

```
VkSwapchainCreateInfo
```

- `surface`
- `imageFormat`
- `imageColorSpace`
- `imageExtent`
- `imageArrayLayers`
- `imageUsage`
- `imageSharingMode`
- `preTransform`
- `compositeAlpha`
- `presentMode`
- `clipped`

```
vkGetDevicePhysicalSurfaceCapabilities
```

```
VkSurfaceCapabilities
```

- `minImageCount`
- `maxImageCount`
- `currentExtent`
- `minImageExtent`
- `maxImageExtent`
- `maxImageArrayLayers`
- `supportedTransforms`
- `currentTransform`
- `supportedCompositeAlpha`

```
vkGetSwapChainImages
```

```
vkCreateImageView
```

```
uint32_t
```

## Present Images

```
PresentImages = new VkImage[imageCount];
```

```
for( unsigned int i = 0; i < imageCount; i++ )
```

```
VkImageViewCreateInfo
```

- `viewType` = VK_IMAGE_VIEW_TYPE_2D
- `format` = VK_FORMAT_B8G8R8A8_UNORM
- `components` = VK_COMPONENT_SWIZZLE_RGBA
- `aspectMask` = VK_IMAGE_ASPECT_COLOR_BIT
- `baseMipLevel` = 0
- `levelCount` = 1
- `baseArrayLayer` = 0
- `layerCount` = 1

```
vkCreateImageView
```

```
VkSemaphore
```

```
vkCreateSemaphore
```

```
uint64_t timeout = UINT64_MAX;
```

```
vkAcquireNextImageKHR
```

```
vkBeginCommandBuffer
```

```
vkCmdBeginRenderPass
```

```
vkCmdBindPipeline
```

```
vkCmdEndRenderPass
```

```
vkEndCommandBuffer
```

Rendering into the Swap Chain, I
Rendering into the Swap Chain, II

```c
VkFenceCreateInfo vfci;
vfci.sType = VK_STRUCTURE_TYPE_FENCE_CREATE_INFO;
vfci.pNext = nullptr;
vfci.flags = 0;
VkFence renderFence;
vkCreateFence( LogicalDevice, &vfci, PALLOCATOR, OUT &renderFence );

VkQueue presentQueue;
vkGetDeviceQueue( LogicalDevice, FindQueueFamilyThatDoesGraphics(), 0, OUT &presentQueue );
...

VkSubmitInfo vsi;
vsi.sType = VK_STRUCTURE_TYPE_SUBMIT_INFO;
vsi.pNext = nullptr;
vsi.waitSemaphoreCount = 1;
vsi.pWaitSemaphores = &imageReadySemaphore;
vsi.pWaitDstStageMask = &waitAtBottom;
vsi.commandBufferCount = 1;
vsi.pCommandBuffers = &CommandBuffers[nextImageIndex];
vsi.signalSemaphoreCount = 0;
vsi.pSignalSemaphores = &SemaphoreRenderFinished;
result = vkQueueSubmit( presentQueue, 1, IN &vsi, IN renderFence ); // 1 = submitCount
```

Rendering into the Swap Chain, III

```c
result = vkWaitForFences( LogicalDevice, 1, IN &renderFence, VK_TRUE, UINT64_MAX );
VkPresentInfoKHR vpi;
vpi.sType = VK_STRUCTURE_TYPE_PRESENT_INFO_KHR;
vpi.pNext = nullptr;
vpi.waitSemaphoreCount = 0;
vpi.pWaitSemaphores = (VkSemaphore *)nullptr;
vpi.swapchainCount = 1;
vpi.pSwapchains = &SwapChain;
vpi.pImageIndices = &nextImageIndex;
vpi.pResults = (VkResult *) nullptr;
result = vkQueuePresentKHR( presentQueue, IN &vpi );
```