Chapter 9: `Rendering` 🎯📜

1. `vkAcquireNextImageKHR()`

- https://vkdoc.net/man/vkAcquireNextImageKHR
  - `.device` = ▨ *Same as* `SwapChain` 👩‍🎨
    - *So, now you know which class this function has got to be inside* 😱
  - `.swapchain` = ▨ 👩‍🎨
  - `.timeout` ⏱️ ⚡ `nanoseconds`
    - *specifies how long the function waits, in* ⏱️ ⚡ `nanoseconds`*, if no image is available.*

      ```
      uint64_t ns_per_second = 1'000'000'000;
      ```

  - `.semaphore` 🔗 *SubChapter 2*
  - `.fench` ▨ *ChapterZZZ*
  - `.pImageIndex` ↩️ 📦
    - *Well, this function doesn't return an* `VkImage` *but an index to it* 👩‍🎨
- 📜 **REY_DOCs**
  - `VK_SUBOPTIMAL_KHR`
    - *if the window has been resized but the OS/platform's* `GPU-DriverImplementation` / `PresentationEngine` *is still able to scale the presented images to the new size to produce valid surface updates.*
    - *It is up to the application to decide whether it prefers to continue using the current swapchain in this state, or to re-create the swapchain to match resized window.*
  - `VK_ERROR_OUT_OF_DATE_KHR`
    - *the images in the swapchain no longer matches the surface properties (e.g., the window was resized)*
    - *and the presentation engine can't present them,*
    - *so the application needs to create a new swapchain that matches the surface properties.*
  - *REFs:- 1. minerva*

---

2. `VkSemaphore` ▩ ChapterZZZ

- https://vkdoc.net/man/VkSemaphore
  - *I wouldn't suggest reading it right now tho* 😅
  - *But, basically,*
    - `SemaPhore` *will be used to synchronize the rendering and presentation of images*

1. `VkSemaphoreCreateInfo`
   - *https://vkdoc.net/man/VkSemaphoreCreateInfo*
     - `.sType` = ▨ `VK_STRUCTURE_TYPE_SEMAPHORE_CREATE_INFO`
     - `.pNext` = ∅ `NULL`
     - `.flags` = ▨ `0`

2. `vkCreateSemaphore`
   - *https://vkdoc.net/man/vkCreateSemaphore*
     - `.device`
     - `.pCreateInfo` ▨ 👩‍🎨
     - `.pAllocator` ▨ *ChapterZZZ*
     - `.pSemaphore` ↩️ 📦

`So far, The result` **4.guide.chapter9.3.swapchain.hh**

3. Command Recording

1. `VkCommandBufferBeginInfo`
   - *https://vkdoc.net/man/VkCommandBufferBeginInfo*
     - `.sType` = `VK_STRUCTURE_TYPE_COMMAND_BUFFER_BEGIN_INFO`
     - `.pNext` = `NULL`
       - `VkDeviceGroupCommandBufferBeginInfo`
     - `.flags` `VkCommandBufferUsageFlagBits`
       - *https://vkdoc.net/man/VkCommandBufferUsageFlagBits* | *ivirtex-github*
         - `ONE_TIME_SUBMIT`
         - `RENDER_PASS_CONTINUE` *[secondary command buffer]*
         - `SIMULTANEOUS_USE`
     - `.pInheritanceInfo` *[secondary command buffer]*

2. `VkRenderPassBeginInfo`
   - *https://vkdoc.net/man/VkRenderPassBeginInfo*
     - `.sType` = `VK_STRUCTURE_TYPE_RENDER_PASS_BEGIN_INFO`
     - `.pNext` = `NULL`
     - `.renderPass` =
     - `.framebuffer` =
     - `.renderArea`
       - *https://vkdoc.net/man/VkRect2D*
     - `.pClearValues`
       - *https://vkdoc.net/man/VkClearValue*

4. `amVK_SurfacePresenter`

Can't have everything scatterred now, everything is getting too much sophisticating.... 🤔 🤦♀ must *Refactor*....

Before Commit:- https://github.com/REYNEP/amGHOST/blob/9cec3e58db123144bd8d88363ccf9a4a7ffc9edc/amVK/amVK_Surface.hh
After Commit:- https://github.com/REYNEP/amGHOST/blob/3be7cfcd154b383cd98783d302468f63fda0618b/amVK/amVK_SurfacePresenter.hh

`🎥 So far, The result` [🔗 `GITHUB`]

- 📝 amVK_SwapChain.hh
- 📝 amVK_SurfacePresenter.hh
- 📝 amVK_Device.hh
- 📝 amVK_DeviceQCI.hh
-