Chapter 2: `VkDevice`

Overview

We need to create/get hold of a couple of handles:

Instance	1 `VkInstance` per program/app	**VkInstance**
Window Surface	Surface(OS-Window) [for actually linking Vulkan-Renders to Screen/Surface]	**VkSurfaceKHR**
Physical Device	An Actual HARDWARE-GPU-device	**VkPhysicalDevice**
Queue	Queue(Commands) to be executed on the GPU	**VkQueue**
Logical Device	The "Logical" GPU Context/Interface (Software Layer)	**VkDevice**
Swap Chain	Sends Rendered-Image to the Surface(OS-Window) Keeps a backup image-buffer to Render onto	**VkSwapchainKHR**

Take a look into this awesome *slide* from slide-26 onwards
...to understand what each of these steps above *"feel like"*/mean/*"how to imagine them"*.
*slide = *Vulkanised 2023 Tutorial Part 1*

0. `amVK` wrap

```cpp
#include "amVK_Instance.hh"
#include "amVK_DeviceQueues.hh"
#include "amVK_Device.hh"


    // TwT
    REY_LOG("");
amVK_Instance::EnumeratePhysicalDevices();
amVK_GPUProps  *GPUProps = amVK_InstanceProps::GetARandom_GPU();
            GPUProps->GetPhysicalDeviceQueueFamilyProperties();
            GPUProps->REY_CategorizeQueueFamilies();


amVK_Device* D = new amVK_Device(GPUProps);
    D->CI                      // VkDeviceCreateInfo      [public]
    D->Queues                  // amVK_DeviceQueues       [public] [take a look inside 🤭]
    D->add_1D_QFAMs_QCount_USER()   // amVK_DeviceQueues
    D->CreateDevice(1);        // param1 = GraphicsQueueCount =
    D->GetDeviceQueues();      //     see:- Queues.TheArrays 🤭
    D->Queues.GraphicsQ(0)     // returns   Queues.TheArrays.Graphics[0]
```

1

1. 🎁 `vkCreateDevice()`

- *https://vkdoc.net/man/vkCreateDevice*
  - `physicalDevice` 🔲 `HardwareGPU_List[0]` / `amVK_InstanceProps::GetARandom_GPU()`
    - *Enumerate* 🔲 *Chapter2.3*
    - *How to 'choose'?* 🔲 *Chapter2.End*
  - `pCreateInfo` 🔲 🧝
    - 🔗 *SubChapter 2*
  - `pAllocator` 🔲 *ChapterZZZ*
  - `pDevice` 🔁 🧊 `&m_Device`
    - 🔁 🧊 *: "Returned by vkFunc()"*

- • We are not gonna call the `vkCreateDevice()` yet....
  - ○ But, yes, we've already made the class container around it 😄
    - ◆ *4.guide.chapter2.2.midway.hh*
  - ○ we'll actually call this functiion in 🟧 *Chapter2.8*
  - ○ Then, Why am I telling you about this now, here?
    - ◆ because, the idea is, our sole task is to *fill it up step by step*
    - ◆ so we did need to know first about `vkCreateDevice()`
      </br>

- 🎥 `So far, The result` :-
  - *4.guide.chapter2.2.midway.hh*

2. 🛠 `VkDeviceCreateInfo`

- *https://vkdoc.net/man/VkDeviceCreateInfo*
  - `.sType` 🔲 `VK_STRUCTURE_TYPE_DEVICE_CREATE_INFO`
  - `.pNext` ⊘ `nullptr`
    - ⊘ *almost any EXT that you are gonna enable.... is prolly gonna end up being passed on here.... tied to* `VkDeviceCI` 🧝
  - `.flags` 🚩 `0`
    - 🚩 *: "No Flag"*
    - `VkSpecs` *Says:-* `reserved for future use`
  - `.pQueueCreateInfos` 🔗 *SubChapter 5*
    - *Multiple Queue Create Infos:-* 🔲 *Chapter2.8*
  - `.ppEnabledLayerNames` ⚠ *deprecated [by Vulkan]*
  - `.ppEnabledExtensionNames` 🔲 *Chapter4.2*
  - `.pEnabledFeatures` 🔲 *ChapterZZZ*
    - *This should be really interesting*
- 📜 *REY_DOCs*
  - `.pQueueCreateInfos` *-> yes, you 'can' pass multiple* 😊
  - *Sometimes there will be* `.zzzCreateInfoCount` & `.pZZZCreateInfos`
    - *So you could like pass in an array/vector*
    - *You will see this in lots of other places*
- 🎥 `So far, The result` :-
  - *4.guide.chapter2.3.midway.hh*

3. 📖 `vkEnumeratePhysicalDevices()`

- *https://vkdoc.net/man/vkEnumeratePhysicalDevices*

- `</> TheCode`

```cpp
uint32_t deviceCount = 0;
    // [implicit valid usage]:- must be 0 [if 3rd-param = nullptr]
    vkEnumeratePhysicalDevices(m_instance, &deviceCount, nullptr);
        // it's kinda like the function is 'output-ing into' deviceCount


std::vector<VkPhysicalDevice> HardwareGPU_List(gpuCount);
    // best to save this as a class member variable
    vkEnumeratePhysicalDevices(m_instance, &deviceCount, HardwareGPU_List.data());
        // note: it does return      VkResult return_code
```

- 👀 `Visualization / [See it] / JSON Printing` :- *4.guide.chapter2.1.json.hh*
- 🎬 `So far, The result` :- *4.guide.chapter2.1.midway.hh*
- 🔗 `GitHub` :- *amVK_GPUProps.hh*

4. ◈ `amVK_InstanceProps::GetARandom_GPU()`

`</> TheCode`   🔗 `GITHUB` *amVK_ InstanceProps.hh#L39*

5. ⚒ `VkDeviceQueueCreateInfo` - *'The Real Deal'*

- *https://vkdoc.net/man/VkDeviceQueueCreateInfo*
    - `.sType` ▨ `VK_STRUCTURE_TYPE_DEVICE_QUEUE_CREATE_INFO`
    - `.pNext` ⊘ `nullptr`
        - ⊘ *2 Extensions* 😊 *(will talk about them later)*
    - `.flags` ☐ `0`
        - ⊘ *https://vkdoc.net/man/VkDeviceQueueCreateFlagBits* | *ivirtex-github*
        - ▶ : *"Only Option"*
            - `VK_DEVICE_QUEUE_CREATE_PROTECTED_BIT` *[Protected Queue]*

    - `.queueFamilyIndex` 🔗 *Next 3 SubChapters*
        - `vkGetPhysicalDeviceQueueFamilyProperties()` *--> look for a QueueFamily that supports* `VK_QUEUE_GRAPHICS_BIT`

    - `.queueCount` ▨ `1` *[Specify, how many you need* 👩‍💻 *]*

    - `.pQueuePriorities` *--> yes, this can be multiple "Priorities"* 😵 *[idk yet why tho]*
        - *Range = (0.0 -> 1.0) [inclusive]*
        - *Within the same device, queues with higher priority may be allotted more processing time than queues with lower priority.*

- 🎬 `So far, The result` :-
    - *We are gonna take a Big Leap & Start connecting to* 🔗 `GITHUB`
    - *amVK_DeviceQCI.hh*

6. 📜 `vkGetPhysicalDeviceQueueFamilyProperties()`

- *https://vkdoc.net/man/vkGetPhysicalDeviceQueueFamilyProperties*
- 📜 *REY_DOCs*
  - ◦ *a GPU can have "multiple QueueFamilies"*
    - ■ *a* `QueueFamily` *might support* `VK_QUEUE_GRAPHICS_BIT`
    - ■ *another* `QueueFamily` *might support* `VK_QUEUE_COMPUTE_BIT`
    - ■ *another* `QueueFamily` *might support* `VK_QUEUE_TRANSFER_BIT`
    - ■ *another* `QueueFamily` *might support* `VK_QUEUE_VIDEO_ENCODE_BIT_KHR`
    - ■ *another* `QueueFamily` *might support a-mixture of multiple*
    - ■ *talking about this in ->* 🔗 *Next SubChapter*

- `</> TheCode` *[OldWay]*

```cpp
#define GPUs                              amVK_InstanceProps::s_HardwareGPU_List
#define amVK_2D_GPUs_QFAMs                amVK_Instance::s_HardwareGPU_QFamProps_List2D
static inline REY_Array<REY_Array<VkQueueFamilyProperties>>   s_HardwareGPU_QFamProps_List2D;
    // REY_Array --> "REY_LoggerNUtils/REY_Utils.hh" 😄
    // 1 System/PC
        // multiple GPU
            // multiple QFamProps


static inline void GetPhysicalDeviceQueueFamilyProperties(void) {
    amVK_2D_GPUs_QFAMs.reserve(GPUs.n);              // malloc using "new" keyword
    for ( uint32_t k = 0;  k < GPUs.n; k++ )        // for each GPU
    {
        REY_Array<VkQueueFamilyProperties> *k_QFamProps = &amVK_2D_GPUs_QFAMs.data[k];

        uint32_t QFamCount = 0;
            vkGetPhysicalDeviceQueueFamilyProperties(GPUs[k], &QFamCount, nullptr);

        k_QFamProps->n = QFamCount;
        k_QFamProps->data = new VkQueueFamilyProperties[QFamCount];
            vkGetPhysicalDeviceQueueFamilyProperties(GPUs[k], &k_QFamProps->n, k_QFamProps->data);
    }
    #undef GPUs
}
```

  - ◦ 👀 `Visualization / [See it] / JSON Printing` *:- 4.guide.chapter2.5.json.hh*
    - ■ *Check the 3070 JSON by REY*
  - ◦ 🎬 `So far, The result` *:- [OldWay] 4.guide.chapter2.5.amVK_Instance.hh*
    - ■ *Compare to -> 4.guide.chapter2.1.midway.hh*
      - ■ `2DArray_QFAM_Props` *part & below were added only compared to* `Chapter2.1.`

- 🎬 `So far, The result` *:- 🔗* `GITHUB` *[NewWay]*
  - ◦ *amVK_GPUProps.hh*
  - ◦ *amVK_GPUProps.cpp#L5-L17*

7. 👈 `VkQueueFamilyProperties`

- *https://vkdoc.net/man/VkQueueFamilyProperties*
- 📜 *REY_DOCs*
    - `.queueFlags`
        - *we are gonna choose a `QCI.queueFamilyIndex` based on these flags*
        - *primarily, for the least, we wanna choose a `QueueFamily` that supports `VK_QUEUE_GRAPHICS_BIT`*
        - *all kinds of amazing things can be done using*
            - `VK_QUEUE_COMPUTE_BIT`
            - `VK_QUEUE_TRANSFER_BIT`
            - `VK_QUEUE_VIDEO_ENCODE_BIT_KHR`
    - `.queueCount`
        - *yes there is a limit to 'how many `Queues` we are allowed to work with'* 😎
    - `.timestampValidBits`
    - `.minImageTransferGranularity`

8. `VkDeviceQCI.queueFamilyIndex` [OldWay]

- 🎯 `Task`
    - *is to choose a `QueueFamily` that supports `VK_QUEUE_GRAPHICS_BIT`* 😵‍💫
    - *(if you've followed on so far -> this should be easy* 😵‍💫 *)*
- `</> amVK_Device.hh`

```cpp
void amVK_Device::Select_QFAM_GRAPHICS(void) {
    if (!amVK_Instance::called_GetPhysicalDeviceQueueFamilyProperties) {
        amVK_Instance::EnumeratePhysicalDevices();
    }

    if (!amVK_Instance::called_GetPhysicalDeviceQueueFamilyProperties) {
        amVK_Instance::GetPhysicalDeviceQueueFamilyProperties();
    }

    amVK_Instance::amVK_PhysicalDevice_Index index = amVK_HEART->GetARandom_PhysicalDevice_amVK_Index();
    this->QCI.Default.queueFamilyIndex = amVK_Instance::ChooseAQueueFamily(VK_QUEUE_GRAPHICS_BIT,
index);
        // If you wanna see the implementation for this function
}
```

- 🎞️ `So far, The result` *:- OldWay (Don't spend time inside this, more than 1 minute)*
    - *4.guide.chapter2.9.Props.hh*
    - *4.guide.chapter2.9.amVK.cpp*

- 🎞️ `So far, The result` *:- NewWay* 🔗 `GITHUB` *(NewWay is like 10x more organized and easier to understand)*
    - *amVK_GPUProps.hh*
    - *amVK_GPUProps.cpp#L266-L286*

9. ◈ `REY_CategorizeQueueFamilies()` [NewWay]

`</> TheCode` 🔗 `GITHUB`
amVK_ GPUProps.hh#L50
amVK_ GPUProps.cpp#L260

10. back to 🎁 `vkCreateDevice()` finally calling it ☺

- *https://vkdoc.net/man/VkDeviceCreateInfo*

- `</> main.cpp`

```cpp
amVK_Device* D = new amVK_Device(GPUProps);
    D->CI                       // VkDeviceCreateInfo        [public]
    D->Queues                   // amVK_DeviceQueues         [public] [take a look inside 🤨]
    D->add_1D_QFAMs_QCount_USER()  // amVK_DeviceQueues
    D->CreateDevice(1);         // param1 = GraphicsQueueCount = 1
    D->GetDeviceQueues();       //     see:- Queues.TheArrays 🤨
    D->Queues.GraphicsQ(0)      // returns   Queues.TheArrays.Graphics[0]
```

- *Think of this as a PSeudoCode / or / check out my code if you wanna*
- `CreateInfo` *=> By default has initial values inside* `amVK_Device`

11. ◈ `amVK_DeviceQueues`

🔗 *amVK_DeviceQueues.hh*

6

▨ eXtras / TheEnd

11. multiple `VkDeviceCreateInfo.pQueueCreateInfos`

- ***VUID-VkDeviceCreateInfo-queueFamilyIndex-02802***
  - *The* `.queueFamilyIndex` *member of each element of* `.pQueueCreateInfos` *must be unique* 👩‍🔧

  - *So, randomly* `push_back()` *ing without any kinda safety* ➡️ *kinda feels absurd.* 👩‍🔧 *doesn't it? …. e.g.*

    ```
    /* ============ REY_LoggerNUtils::REY_Utils.hh ============ */
    REY_ArrayDYN<VkDeviceQueueCreateInfo> Array = REY_ArrayDYN<VkDeviceQueueCreateInfo>(2);
        REY_ARRAY_PUSH_BACK(Array) = this->Default_QCI;
        REY_ARRAY_PUSH_BACK(Array) =         Your_QCI;
    ```

    - [OldWay]:- *amVK_DeviceQCI.hh*
  - *So what i did is:- to introduce a* `QCount` *array as per* `QFamily` 👩‍🔧
    - [NewWay]:- *amVK_DeviceQueues.hh#L56*
  - *& then have a function for the user to increase the* `QCount`
    - [NewWay]:- 🔗 `GITHUB_WIP` --> `amVK_Device::add_1D_QFAMs_QCount_USER()`

12. *OldWay* 📅 `March, 2025`

  - i. `class amVK_InstanceProps`
    - `EnumeratePhysicalDevices()`
    - `GetPhysicalDeviceQueueFamilyProperties()`
  - *(Don't spend time inside this, more than 1 minute)*
    - 📄 *4.guide.chapter2.9.Props.hh*
    - 📄 *4.guide.chapter2.9.amVK.cpp*
  - *https://github.com/REYNEP/amGHOST/tree/3e44b982902a3f3fa4ac584aefb19da3d4cdfcc6*

13. *NewWay* 📅 `May, 2025`

  - 🔗 `GITHUB` *(NewWay is like 10x more organized and easier to understand)*
    - 📄 *amVK_GPUProps.hh*
    - 📄 *amVK_GPUProps.cpp#L266-L286*

14. `vkGetPhysicalDeviceProperties()` ▨ *Chapter11*
15. *GetFeatures* ▨ *Chapter11*
16. *MemoryTypes* ▨ *Chapter11*
17. *Guide on* `amVK_Array` ▨ *Chapter6.6*