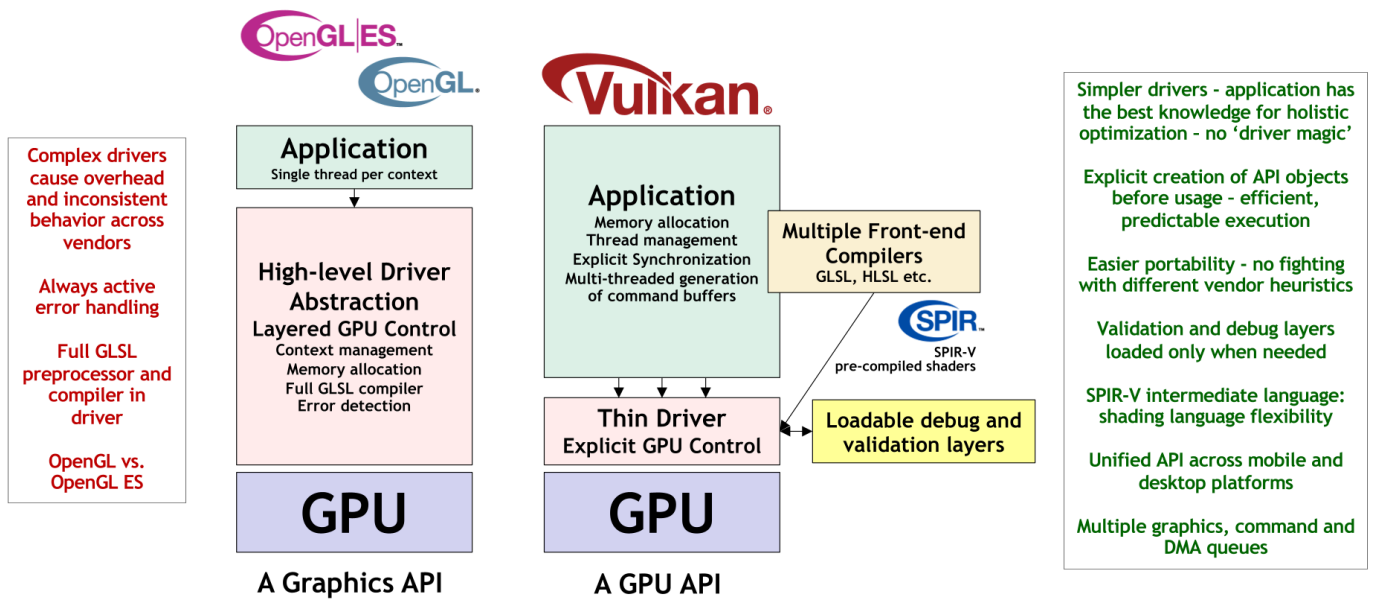


The Real "Adventure" begins here!

[well, not really. I believe the real adventure is in SHADERS and Algorithms!]

Vulkan Explicit GPU Control



© Khronos® Group Inc. 2019 - Page 36

Chapter 1: VkInstance

0. amVK wrap 📱

```
#include "amVK_Instance.hh"

// TwT

amVK_Instance::AppInfo           // VkApplicationInfo           [public]
amVK_Instance::CI                // VkInstanceCreateInfo       [public]

// You can modify these as you wish 😊

amVK_Instance::CreateInstance(); // initializes amVK_HEART
```

1. Notes on 'Notes'

2. VkApplicationInfo

- <https://vkdoc.net/man/VkApplicationInfo>
 - `.sType` --> `VK_STRUCTURE_TYPE_APPLICATION_INFO`
 - `.pNext` --> `NULL`
 - `.pApplicationName` --> null-terminated UTF-8 string
 - `.applicationVersion` --> `uint32`
 - `.pEngineName` --> null-terminated UTF-8 string
 - `.engineVersion` --> `uint32`
 - `.apiVersion` --> `uint32`

• REY_DOCS

- `.apiVersion`
 - **lowest Vulkan API version** Your APP "can run" on.
 - [*clarification needed:- lowest or highest]
- `.engineVersion`
 - and the **version** of the **engine** (if any) used to create "Your APP".
 - This can help **vulkan driver implementations** to perform "ad-hoc" optimizations.
 - e.g. like if a Triple-A [AAA] game used, for say, **Unreal Engine Version 4.1.smth** idk 🤖
- REFS:- [1. minerva](#)

So the first thingy is gonna be the link to the Documentation website 🤖 for the `VkStruct`

Under that,

there's gonna be items/elements of that `VkStruct`

-> Tried to keep them Short & Sorted as per the `vulkan.h` header Declaration

Now I won't copy paste literally every element all the time 🤖

`.sType` & `.pNext` is common
(explained them below)

do remember to check the **Valid Usage** section 🤖 in [vkdoc.net](#)

(i kinda always check that section first, before reading other parts / diving deep)

Sometimes





these items/elements/members




are gonna need some explanation 🤖

-> That's exactly why this `REY_DOCS` section exists!

made with [affine.pro](#) [+ Screenshot of my `4.guide.CH0.pdf`]

• <https://vkdoc.net/man/VkApplicationInfo>

- `.sType` :-
 -  almost every `VkStruct` is gonna have this field/member 🤖
 - must be
 - `VK_STRUCTURE_TYPE_APPLICATION_INFO` for `VkApplicationInfo`
 - `VK_STRUCTURE_TYPE_INSTANCE_CREATE_INFO` for `VkInstanceCreateInfo`
 - `VK_STRUCTURE_TYPE_DEVICE_CREATE_INFO` for `VkDeviceCreateInfo`
 - and so on... (you get the idea)
- `.pNext` :-
 -  almost every `VkStruct` is gonna have this field/member 🤖
 - must be `NULL`
 - for most of the `VkStruct` s it iss kinda `NULL` 🤖
 - but it has an interesting use case:-
 - <https://vkdoc.net/man/VkDeviceCreateInfo#VUID-VkDeviceCreateInfo-pNext-pNext>
 - you can kinda like pass in pointer to `VkStructEXT` when you need those Extension features 🤖
- `.flags` :-
 -  almost every `VkStruct` is gonna have this field/member 🤖
 - This is not an element of `VkApplicationInfo` per say
 - But, still, i shall list it here....
 - now, sometimes,
 - you will see,  ("Orange Card") [meaning of it below]

- `.pApplicationName` --> null-terminated UTF-8 string
 - `.applicationVersion` --> `uint32`
 - you as the developer of your application can set it to arbitrarily anything you want it to 🤖, say
 - 101
 - 005
 - 1
 - 2025
 - `.pEngineName` --> null-terminated UTF-8 string
 - `.engineVersion` --> `uint32`
 - `.apiVersion` --> `uint32`
- again.... yeah, do remember to check the `Valid Usage` section 😊
 - There's an alternative to [vkdoc.net](https://github.com/ivirtex/vulkan-hover-docs/tree/master/vscode_ext/vulkan_man_md_pages/VkInstanceCreateFlagBits.md)
 - https://github.com/ivirtex/vulkan-hover-docs/tree/master/vscode_ext/vulkan_man_md_pages/VkInstanceCreateFlagBits.md
 - it is also available as an extension in `vscode` --> `ivirtex.vulkan-hover-docs`
 - Symbols
 - :- kinda means nothing
 - i kinda used to like make it look like a bit pattern-ish iguess 🤖🤖
 - :- "Yellow Card"
 - it means, you don't need to hesitate about this thingy right now 🤖 we will focus on this element later 🤖
 - :- "Orange Card"
 - it means, this element is probably never gonna be 'necessary' for vulkan applications 🤖
 - [The extended list can be found in [🔗 Chapter3.14](#)]

2. VkApplicationInfo

- <https://vkdoc.net/man/VkApplicationInfo>
 - `.sType` = `VK_STRUCTURE_TYPE_APPLICATION_INFO`
 - `.pNext` = `NULL`
 - `.pApplicationName` --> null-terminated UTF-8 string
 - `.applicationVersion` --> `uint32`
 - `.pEngineName` --> null-terminated UTF-8 string
 - `.engineVersion` --> `uint32`
 - `.apiVersion` --> `uint32`
- **REY_DOCS**
 - `.apiVersion`
 - lowest Vulkan API version Your APP "can run" on.
 - [*clarification needed:- lowest or highest]
 - `.engineVersion`
 - and the version of the engine (if any) used to create "Your APP".
 - This can help vulkan driver implementations to perform "ad-hoc" optimizations.
 - e.g. like if a Triple-A [AAA] game used, for say, Unreal Engine Version 4.1.smth idk 🤖
 - REFS:- 1. minerva
- yes, what are you waiting for 🤖👩 go go, shooo.... (💎)
 - i. `#include <vulkan/vulkan.h>`
 - ii. take an instance of that `Struct` -> Fill it up [🤖][have the vkdoc.net as assist]

3. VkInstanceCreateInfo

- <https://vkdoc.net/man/VkInstanceCreateInfo>
 - `.sType` = `VK_STRUCTURE_TYPE_INSTANCE_CREATE_INFO`
 - `.pNext` = `NULL`
 - or some interesting Extensions 🤖 (will talk about them later)
 - `.flags` --> `VkInstanceCreateFlagBits`
 - <https://vkdoc.net/man/VkInstanceCreateFlagBits> | ivirtex-github
 - `.pApplicationInfo` --> `ApplicationInfo` 🤖 Duh!
 - `.ppEnabledLayerNames` --> `LayerName` ChapterZZZ
 - `.ppEnabledExtensionNames` --> `ExtensionName` Chapter4.2
 - Don't hesitate about `EnabledLayer` & `EnabledExtensions` right now
 - come back and add them when you need to 🤖
 - This is what I would mean, when i would point smth to a later chapter
 - I will add the 🟡 ("Yellow Card") too!
- **REY_DOCS**
 - Nothing that I need to add, in this section
 - Tho if this section gets big, I will create a separate `.md` file for that thingy

4. A 🧡 Cool `vscode` / `visual-studio` extension if you want 👤

- <https://github.com/ivirtex/vulkan-hover-docs>
 - `vscode` --> `ivirtex.vulkan-hover-docs`
-

5. `VkInstance m_instance = nullptr;`

- <https://vkdoc.net/man/VkInstance>

6. `vkCreateInstance(CI, nullptr, &m_instance)`

- <https://vkdoc.net/man/vkCreateInstance>
 - `param pCreateInfo` = 📄 🧠 Duh!
 - `param pAllocator` = 📄 `nullptr`
 - `param pInstance` = 📄 `&m_instance`
 - 📖 REY_DOCS
 - `param pAllocator` = 📄 `nullptr`
 - `VkAllocationCallbacks` 📄 ChapterZZZ
 - I will make a chapter on this 🧠 [<https://vkdoc.net/chapters/memory#memory-allocation>]
 - Vulkan provides applications the opportunity to perform host memory allocations
 - If this feature is not used
 - the implementation will perform its own memory allocations.
 - Since most memory allocations are off the critical path, this is not meant as a performance feature. Rather, this can be useful for certain embedded systems, for debugging purposes (e.g. putting a guard page after all host allocations), or for memory allocation logging.
-

7. 🛑 Error Handling / Checking / 💎 Logging

- check out my `amVK_log.hh`
 - uses [REY_LoggerNUtils](#) inside `amGHOST`
 - has a simple `stackTracer()` that i basically stripped from blender3D codebase 🙄

8. 📦 So far, The result

- [4.guide.chapter1.hh](#)
-

9. The Unused ones

1. `vkEnumerateInstanceExtensionProperties()` --> 📄 Chapter4.2
 - <https://vkdoc.net/man/vkEnumerateInstanceExtensionProperties>
2. `Add_InstanceEXT_ToEnable(const char* extName)` --> 📄 Chapter4.2
 - this is a `amVK/REY Custom Function`