Chapter 4: `VkSwapchainKHR` ⚠

0. `VkSwapchainCreateInfoKHR` ℹ

- https://vkdoc.net/man/VkSwapchainCreateInfoKHR
  - `.flags` -> *"ChapterZZZ"*
  - `.surface` -> *next part [Chapter4.2]*
  - *image options -> next part [Chapter4.4]*
    - `.minImageCount` ->
    - `.imageFormat` -> 😭
    - `.imageColorSpace` -> 😭
    - `.imageExtent` -> 😊
    - `.imageArrayLayers`
    - `.imageUsage`
    - `.imageSharingMode` -> `EXCLUSIVE/CONCURRENT` *[Toggle]*
  - `VK_SHARING_MODE_CONCURRENT` -> *"ChapterZZZ"*
    - `.queueFamilyIndexCount` -> *if using, must be* `greated than 1`
    - `.pQueueFamilyIndices`
  - *more image options -> next part*
    - `.preTransform` :- `VkSurfaceTransformFlagBitsKHR`
    - `.compositeAlpha` :- `VkCompositeAlphaFlagBitsKHR`
    - `.presentMode` :- `VkPresentModeKHR`
    - `.clipped` :- `VkBool32`
  - `.oldSwapchain` -> *"ChapterZZZ"*

1. `amVK` wrap 1

```
// TwT

    REY_LOG("");
amVK_GlobalProps::EnumerateInstanceExtensions();
amVK_Instance::Add_InstanceEXT_ToEnable("VK_KHR_surface");
amVK_Instance::Add_InstanceEXT_ToEnable(amGHOST_System::get_vulkan_os_surface_ext_name());
    // amGHOST_VkSurfaceKHR::create_surface() needs that extension enabled
amVK_Instance::CreateInstance();

    REY_LOG("");
VkSurfaceKHR  VK_S = amGHOST_VkSurfaceKHR::create_surface(W, amVK_Instance::vk_Instance);

// another amVK_Wrap, at the end of this file
```

2. `VkSurfaceKHR` 🏄♀️

- https://vkdoc.net/man/VkSurfaceKHR
- https://vkdoc.net/extensions/VK_KHR_surface
  - *Yaaaay, we have reached our first extension to enable*
    - *we need to enable it back in* `vkCreateInstance()` *from* ***Chapter1.2***

1. `vkEnumerateInstanceExtensionProperties()`
   - *https://vkdoc.net/man/vkEnumerateInstanceExtensionProperties*
   - *Implement Exactly like* ***Chapter2.1*** 😉
     - `vkEnumeratePhysicalDevices()`
2. `IS_InstanceEXT_Available(const char* extName)`

```
bool amVK_GlobalProps::IS_InstanceEXT_Available(const char *extName) {
    for (uint32_t k = 0, lim = amVK_EXT_PROPs.n; k < lim; k++) {
        if (strcmp(amVK_EXT_PROPs[k].extensionName, extName) == 0) {    // <cstring>
            return true;
        }
    }
    return false;
}
```

3. `Add_InstanceEXT_ToEnable(const char* extName)`

```
static inline REY_ArrayDYN<char*> s_Enabled_EXTs = REY_ArrayDYN<char*>(nullptr, 0, 0);
    // It will be automatically allocated, resize, as we keep adding 😊
#include <string.h>
void amVK_Instance::Add_InstanceEXT_ToEnable(const char* extName)
{
    if (!amVK_GlobalProps::called_EnumerateInstanceExtensions) {
         amVK_GlobalProps::EnumerateInstanceExtensions();
    }

    if (amVK_GlobalProps::IS_InstanceEXT_Available(extName)) {
        char  *dont_lose = new char[strlen(extName)];
        strcpy(dont_lose, extName);

        s_Enabled_EXTs.push_back(dont_lose);

        amVK_Instance::CI.enabledExtensionCount = s_Enabled_EXTs.neXt;
        amVK_Instance::CI.ppEnabledExtensionNames = s_Enabled_EXTs.data;
    }
    else {
        REY_LOG_notfound("Vulkan Extension:- " << extName);
    }
}
```

4. OS Specfic SurfaceEXT & Creating it

```
amVK_Instance::Add_InstanceEXT_ToEnable(amGHOST_System::get_vulkan_os_surface_ext_name());
```

```
    // or
amVK_Instance::Add_InstanceEXT_ToEnable("VK_KHR_win32_surface");
    // or some other surface name
```

i. `VkWin32SurfaceCreateInfoKHR` & `vkCreateWin32SurfaceKHR()`
   - *https://vkdoc.net/man/VkWin32SurfaceCreateInfoKHR*

```
pure-virtual VkSurfaceKHR amGHOST_VkSurfaceKHR_WIN32::create(VkInstance I)
{
    amGHOST_SystemWIN32 *heart_win32 = (amGHOST_SystemWIN32 *) amGHOST_System::heart;
    VkWin32SurfaceCreateInfoKHR CI = {
        .sType = VK_STRUCTURE_TYPE_WIN32_SURFACE_CREATE_INFO_KHR,
        .pNext = NULL,
        .flags = 0,
        .hinstance = heart_win32->_hInstance,
        .hwnd = this->W->m_hwnd
            // W = amGHOST_WindowWIN32
    };

    VkSurfaceKHR S = nullptr;
    VkResult return_code = vkCreateWin32SurfaceKHR(I, &CI, nullptr, &S);
    amVK_return_code_log( "vkCreateWin32SurfaceKHR()" );
    return S;
}
```

ii. `VkXlibSurfaceCreateInfoKHR` & `vkCreateXlibSurfaceKHR()` 🛠 *[wip]*

iii. **REY_DOCs**
   - *you can also check* `amGHOST_VkSurfaceKHR::create_surface()` 😉

iv. **So far, The result:- 4.guide.chapter4.2.TheEnd.hh**
   - *in the end people will just use 1 line*

```
VkSurfaceKHR VK_S = amGHOST_VkSurfaceKHR::create_surface(amG_WindowOBJ, amVK_Instance::s_vk);
```

---

3. `Naming Patterns` ✒

- example naming patterns for storing all these data.... cz *it's gonna get overwhelming pretty soon, pretty fast*

1. `Arrays`

```
class amVK_GlobalProps {
    public:
        // Array of `HardWare amVK_1D_GPUs` connected to motherboard
    static inline REY_Array<VkPhysicalDevice>                          amVK_1D_GPUs;
    static inline REY_Array<REY_Array<VkQueueFamilyProperties>>        amVK_2D_GPUs_QFAMs;
    static inline REY_Array<VkExtensionProperties>                     amVK_1D_InstanceEXTs;
```

```
    static inline REY_ArrayDYN<char*>
amVK_1D_InstanceEXTs_Enabled;
    static inline REY_ArrayDYN<SurfaceInfo>                                amVK_1D_SurfaceInfos;
    static inline REY_Array<REY_Array<VkExtensionProperties>>              amVK_2D_GPUs_EXTs;
        // REY_Array  doesn't allocate any memory by default

    #define amVK_LOOP_GPUs(_var_)           \
        for (uint32_t _var_ = 0, lim = amVK_1D_GPUs.n;             _var_ < lim;  _var_++)
    #define amVK_LOOP_QFAMs(_k_, _var_)     \
        for (uint32_t _var_ = 0, lim = amVK_2D_GPUs_QFAMs[_k_].n;  _var_ < lim;  _var_++)
};
```

2. `ChildrenStructs`

```
class amVK_GlobalProps {
    public:
    /**
     * VULKAN-EXT:- `VK_KHR_surface`
        *       IMPL:- `amVK_1D_SurfaceInfos`
        */
    class SurfaceInfo {
        public:
        VkSurfaceKHR S = nullptr;
        SurfaceInfo(void) {}
        SurfaceInfo(VkSurfaceKHR pS) {this-> S = pS;}

                REY_Array<REY_Array<VkSurfaceFormatKHR>>          amVK_2D_GPUs_ImageFMTs;

        bool called_GetPhysicalDeviceSurfaceFormatsKHR = false;
        void        GetPhysicalDeviceSurfaceFormatsKHR(void);  // amVK_2D_GPUs_ImageFMTs
    };
};
```

3. `VkFuncCalls`

```
class amVK_GlobalProps {
    public:
    static inline bool called_EnumeratePhysicalDevices = false;
    static inline bool called_GetPhysicalDeviceQueueFamilyProperties = false;
    static inline bool called_EnumerateInstanceExtensions = false;

    public:
    static void EnumeratePhysicalDevices(void);                        // amVK_1D_GPUs
    static       void GetPhysicalDeviceQueueFamilyProperties(void);    // amVK_2D_GPUs_QFAMs
    static void EnumerateInstanceExtensions(void);                     // amVK_1D_InstanceEXTs
};
```

- **REY_DOCs**
  - *Lots of other nice stuffs are happening inside* `amVK_GlobalProps.hh`
- **So far, The result:-**
  - *4.guide.chapter4.3.Props.hh*

- *4.guide.chapter4.3.Props.cpp*
- *4.guide.chapter4.3.PropsOLD.hh*

---

4. `SwapChain Image Options` 🖼

1. `vkGetPhysicalDeviceSurfaceFormatsKHR()`
   - *https://vkdoc.net/man/vkGetPhysicalDeviceSurfaceFormatsKHR*
     - `param surface`
   - **REY_DOCs**
     - *Implement Exactly like Chapter2.5* 😉
       - `vkGetPhysicalDeviceQueueFamilyProperties()`
       - *Only difference is,* `Formats` *might be a bit different as per* `VkSurfaceKHR`
2. `VkSurfaceFormatKHR`
   - *https://vkdoc.net/man/VkSurfaceFormatKHR*
   - **REY_DOCs**
     - *Combo of* `ImageFormat` & `ColorSpace`
       - *so, the gpu kinda expects you to respect these combos, instead of mumbo-jumbo-ing & mixing random stufs alltogether....*
       - *altho, even if you do so, gpu is probably gonna show you the result of WRONG COLORSPACE/IMAGEFORMATs on the screen*

---

3. `Life is Hard without Images/Visualization`
   - *So we are gonna Export to JSON/YAML*
   - *4.guide.chapter4.4.3.Enum2String.hh*
   - *4.guide.chapter4.4.3.data.jsonc*
   - *4.guide.chapter4.4.3.Export.cpp*
     - *dw, don't use this code, it will be refactored & organized in Chapter4.4.6*

---

4. `VkSurfaceCapabilitiesKHR`
   - *https://vkdoc.net/man/VkSurfaceCapabilitiesKHR*
   - **REY_DOCs**
     - `.minImageCount`
       - *2DriverIMPL:- must be at least 1*
     - `.currentExtent`
       - *as the OS Window size changes,* `SurfCaps` *also change*
       - *call* `vkGetPhysicalDeviceSurfaceCapabilitiesKHR()` *to get updated* `WindowSize` / `SurfCaps`
     - `.maxImageArrayLayers`
       - *2DriverIMPL:- must be at least 1*
     - `.supportedTransforms`
       - *2DriverIMPL:- at least 1 bit must be set.*
     - `.supportedUsageFlags`
       - *2DriverIMPL:-* `VK_IMAGE_USAGE_COLOR_ATTACHMENT_BIT` *must be included in the set. Implementations may support additional usages.*
     - `.supportedCompositeAlpha`
       - `ALPHA-Blending/Transparency/GlassEffect` *:- you'd have to enable blending/transparency @ OS-Level first, iguess* 🤔
       - *Transparency -> "ChapterZZZ"*
5. `vkGetPhysicalDeviceSurfaceCapabilitiesKHR()`
   - *https://vkdoc.net/man/vkGetPhysicalDeviceSurfaceCapabilitiesKHR*
   - **REY_DOCs**
     - *we add on top of Chapter4.4.1* 😉
       - `vkGetPhysicalDeviceSurfaceFormatsKHR()`
     - *4.guide.chapter4.4.5.midway.cpp*

6. `Life is Hard without Images/Visualization 2`
   - *Soooooo many things to keep track of, So here we go again*
   - *4.guide.chapter4.4.6.Export.cpp*
   - *4.guide.chapter4.4.6.data.jsonc*

---

7. `VkSharingMode`
   - *https://vkdoc.net/man/VkSharingMode*
   - *it's like a Toggle/Button ->* `EXCLUSIVE/CONCURRENT`
8. **So far, The result:-**

```
amVK_SwapChain *SC = new amVK_SwapChain(VK_Surface);
    SC->CI.imageFormat = VK_FORMAT_B8G8R8A8_UNORM;
    SC->CI.imageColorSpace = VK_COLOR_SPACE_SRGB_NONLINEAR_KHR;
    SC->CI.minImageCount    =
amVK_GlobalProps::amVK_1D_SurfaceInfos[0].amVK_1D_GPUs_SurfCAP[0].minImageCount;
    SC->CI.imageExtent      =
amVK_GlobalProps::amVK_1D_SurfaceInfos[0].amVK_1D_GPUs_SurfCAP[0].currentExtent;
    SC->CI.imageArrayLayers =
amVK_GlobalProps::amVK_1D_SurfaceInfos[0].amVK_1D_GPUs_SurfCAP[0].maxImageArrayLayers;
        // You can just use "1" too, which is guranteed by DRIVER_IMPLEMENTATION [2DriverIMPL]
    SC->CI.imageSharingMode = VK_SHARING_MODE_EXCLUSIVE;
        // `EXCLUSIVE/CONCURRENT` [Toggle]
    SC->CI.imageUsage       = VK_IMAGE_USAGE_COLOR_ATTACHMENT_BIT;
        // 2DriverIMPL:- VK_IMAGE_USAGE_COLOR_ATTACHMENT_BIT is guranteed to be supported by
SurfCAP
```

---

9. `Abbreviations`
   - `PD` -> *PhysicalDevice*
   - `GPUs` -> *PhysicalDevices*
   - `CI` -> *CreateInfo*
   - `QCI` -> *QueueCreateInfo*
   - `QFAM` -> *QueueFamily*
   - `SurfCAP` -> *https://vkdoc.net/man/VkSurfaceCapabilitiesKHR*
   - `SurfFMT` -> *https://vkdoc.net/man/VkSurfaceFormatKHR*
   - `SC` -> *SwapChain*

---

10. `VkSwapchainCreateInfoKHR`

- https://vkdoc.net/man/VkSwapchainCreateInfoKHR
  - `.flags` -> *"ChapterZZZ"*
  - `.surface` -> *Chapter4.2* `VkSurfaceKHR` 💪
  - *image options -> Chapter4.4*
    - `.minImageCount` -> 😊 `SurfCAP.minImageCount`
    - `.imageFormat` -> 😊 `SurfFMT[x].format`
    - `.imageColorSpace` -> 😱 `SurfFMT[x].colorSpace`
      - *Choosing a Combo -> "ChapterZZZ"*
      - *Compositing & ColorSpaces -> "ChapterZZZ"*
    - `.imageExtent` -> 😊 `SurfCAP.minImageCount`
    - `.imageArrayLayers` -> `1`
      - *DriverGurantee*
    - `.imageUsage` -> `VK_IMAGE_USAGE_COLOR_ATTACHMENT_BIT`
      - *DriverGurantee*
    - `.imageSharingMode` -> `EXCLUSIVE/CONCURRENT` *[Toggle]*
      - `VK_SHARING_MODE_CONCURRENT` -> *"ChapterZZZ"*
        - *we aren't gonna use concurrent for now*
        - `.queueFamilyIndexCount` -> `0`
        - `.pQueueFamilyIndices` -> `nullptr`

---

5. `SwapChain Compositing Options` 🔰♂️

1. `.compositeAlpha`
   - *https://vkdoc.net/man/VkCompositeAlphaFlagBitsKHR*
   - **REY_DOCs**
     - `Options` *:- Don't use / Pre-multiplied / Post-multiplied / inherit from OS-native window system*
     - `Requirement` *:-*
       - *You would have to enable @ OS level first, to enable ALPHA/Transparency/GlassEffect for window-s/surfaces*
       - *then after that, if you query for* `vkGetPhysicalDeviceSurfaceCapabilitiesKHR()`
         - `SurfCAP.supportedCompositeAlpha` *will change*
       - *by default, it's prolly always gonna support*
         - `VK_COMPOSITE_ALPHA_OPAQUE_BIT_KHR`
         - *i.e. if you haven't done any mastery wizardry yet, to enable ALPHA/Transparency/GlassEffect*
2. `.preTransform`
   - *https://vkdoc.net/man/VkSurfaceTransformFlagBitsKHR*
   - **REY_DOCs**
     - 🔗 `SurfCAP.currentTransform`
     - *you should probably log it if* `currentTransform` *isn't*
       - `VK_SURFACE_TRANSFORM_IDENTITY_BIT_KHR`
3. `.clipped`
   - **REY_DOCs**
     - *Setting clipped to* `VK_TRUE` *allows the implementation to discard rendering outside of the surface area*
4. `.presentMode` 🏷 `VkPresentModeKHR`
   - *https://vkdoc.net/man/VkPresentModeKHR*

- **REY_DOCs**
  - `Options` :- *IMMEDIATE / MAILBOX / FirstInFirstOut / FIFO_Relaxed*

5. `.oldSwapChain`
   - **REY_DOCs**
     - *if you are "re-creating" swapchain & you had an oldSwapchain*
     - *We do this when*
       a. *Window Size / WindowExtent / Surface was Changed*

6. **So far, The result:-**

```
amVK_SwapChain *SC = new amVK_SwapChain(VK_Surface);
    ... Image Stuffs
    SC->CI.compositeAlpha    = VK_COMPOSITE_ALPHA_OPAQUE_BIT_KHR;
    SC->CI.preTransform      =
amVK_GlobalProps::amVK_1D_SurfaceInfos[0].amVK_1D_GPUs_SurfCAP[0].currentTransform;
    SC->CI.clipped           = VK_TRUE;
    SC->CI.presentMode       = VK_PRESENT_MODE_FIFO_KHR;
    SC->CI.oldSwapchain      = nullptr;
```

6. `SwapChain Extension Enabling` ⚠ [`VK_KHR_swapchain`]

1. `vkEnumerateDeviceExtensionProperties()`
   - *https://vkdoc.net/man/vkEnumerateDeviceExtensionProperties*
     - *honestly this should be named* `vkEnumeratePhysicalDeviceExtensionProperties()`
     - *bcz,*
       - *it doesn't associate with* `VkDevice`
       - *but rather with* `VkPhysicalDevice`
   - **REY_DOCS**

```
class amVK_GlobalProps {
    ...
    static inline bool called_EnumerateDeviceExtensionProperties = false;
    static void               EnumerateDeviceExtensionProperties(void); // amVK_2D_GPUs_EXTs

    static inline REY_Array<REY_Array<VkExtensionProperties>>              amVK_2D_GPUs_EXTs;
    #define amVK_LOOP_GPU_EXTs(_k_, _var_)  for (uint32_t _var_ = 0,  lim =
amVK_2D_GPUs_EXTs[_k_].n;   _var_ < lim;  _var_++)

    static bool      IS_GPU_EXT_Available(PD_Index GPU_k, const char *extName); //
amVK_2D_GPUs_EXTs
    // kinda copy of IS_InstanceEXT_Available
    ...
};
```

2. `amVK_Device::Add_GPU_EXT_ToEnable(const char* extName)`

```
class amVK_Device {
    ...
```

```
    REY_ArrayDYN<char*>   amVK_1D_DeviceEXTs_Enabled;
    void Log_GPU_EXTs_Enabled(VkResult ret);
    void Add_GPU_EXT_ToEnable(const char* extName);
        // Copy of `amVK_GlobalProps::Add_InstanceEXT_ToEnable()` -> but not static anymore....
};
```

3. So far, The result:-
   - *4.guide.chapter4.6.newStuffs.hh*
   - *4.guide.chapter4.7.Props.hh*
   - *4.guide.chapter4.7.Props.cpp*

7. `vkCreateSwapchainKHR()`

- https://vkdoc.net/man/vkCreateSwapchainKHR
- [TODO]:- Add the commit-tree Link
- It took me 5days to complete *Chapter4*
  - *(well, i worked on a houdini project for 2 days.... so yeah)*

8. `amVK` wrap 2

```
{
    amVK_GlobalProps::EnumerateDeviceExtensionProperties();

    amVK_Device* D = new amVK_Device(amVK_GlobalProps::GetARandom_GPU());
        D->select_QFAM_Graphics();
        D->Add_GPU_EXT_ToEnable("VK_KHR_swapchain");
        D->CreateDevice();
}
```

9. `amVK` wrap 3

```
    // TwT

        REY_LOG("")
    amVK_Surface   *S  = new amVK_Surface(VK_S);
    amVK_Presenter *PR = S->PR;
                           PR->bind_Device(D);
                           PR->create_SwapChain_interface();
                               // This amVK_SwapChain is Bound to this amVK_Surface
    amVK_SwapChain *SC =    PR->SC;
        SC->konf_ImageSharingMode(VK_SHARING_MODE_EXCLUSIVE);
        SC->konf_Images(
            amVK_IF::RGBA_8bpc_UNORM,    // VK_FORMAT_R8G8B8A8_UNORM
            amVK_CS::sRGB,               // VK_COLOR_SPACE_SRGB_NONLINEAR_KHR
            amVK_IU::Color_Display       // VK_IMAGE_USAGE_COLOR_ATTACHMENT_BIT
        );
```

```
        SC->konf_Compositing(
            amVK_PM::FIFO,              // VK_PRESENT_MODE_FIFO_KHR
            amVK_CC::YES,               // Clipping:- VK_TRUE
            amVK_TA::Opaque             // VK_COMPOSITE_ALPHA_OPAQUE_BIT_KHR
        );
        SC->sync_SurfCaps();            // refresh/fetch & set/sync ---> latest SurfCaps

        SC->CI.oldSwapchain     = nullptr;
        SC->CreateSwapChain();
```