6. `amVK_SurfacePresenter`

Can't have everything scatterred now, everything is getting too much sophisticating.... 🤓 🥷 must *Refactor*....

Major Decision Change

Right now, `amVK_Surface::CTOR` creates `amVK_SurfacePresenter` . & `SwapChain`, `RenderPass`, `CommandPool` are supposed to be created from `amVK_SurfacePresenter` .

```
class amVK_Surface
    amVK_SurfacePresenter {
        create_SwapChain_interface()
            new amVK_SwapChain(this)
                this->CI.surface = PR->S->vk_SurfaceKHR;
                // later amVK_SwapChain::CreateSwapChain(void) uses this->PR->D->vk_Device
        create_RenderPass_interface()
            new amVK_RenderPass(this)
                this->PR = PR;
        create_CommandPool_interface()
            new amVK_CommandPool(this)
                this->CI.queueFamilyIndex = this->PR->D->amVK_1D_QCIs.ptr_Default()->queueFamilyIndex;
        create_FrameBuffers()
            new amVK_FrameBuffer(this)
                this->CI.renderPass = this->PR->RP->vk_RenderPass;
```

Problem #1:- I think this is just a little too much deep to handle....
Problem #2:- if `amVK_SwapChain.hh` included `amVK_SurfacePresenter.hh` , then the reverse can't happen. 🙍
Thus a lot of 1-liner functions would have to be put inside `.cpp` even tho i don't want it to.

1. `Problem #2:- in Details`

- *amVK_SurfacePresenter.hh#L37*
- *amVK_SwapChain.hh#L48*
- *The Solution*
  - `C1` :- *Don't include* `amVK_SurfacePresenter.hh` *in* `amVK_SwapChain.hh` *but rather inside* `amVK_SwapChain.cpp`
  - `C2` :- *Don't include* `amVK_SwapChain.hh` *in* `amVK_SurfacePresenter.hh` *but rather inside* `amVK_SurfacePresenter.cpp`
- `Case 1` :-
  - `amVK_SwapChain::CONSTRUCTOR`
  - `sync_SurfCaps()`
  - *both of these have to go inside* `amVK_SwapChain.cpp`
- `Case 2` :-
  - `amVK_SurfacePresenter::sync_SC_SurfCaps()`
  - `amVK_SurfacePresenter::synced_ImageExtent()`
  - *both of these (& as of my plan right now, heck ton of other 1 liner function) are gonna have to go inside* `amVK_SurfacePresenter.cpp`

2. *Weeelll*

- *There is one other solution.... That is to change the design.... Which is what I figured is should do.... Not everybody would want to use* `amVK_SurfacePresenter` *anyway* 🙍
- *2 Ways:-*
i. *Making* `amVK_SurfacePresenter` *Optional*
  a. *None of the other amVK_Class is gonna depend on this anymore*
  b. *amVK_SurfacePresenter serving as like a top level NODETREE system with extra PRESET Functions / soo. (If you are looking from a NodeEditor perspective)*
  c. *This is like having a BIG BAD NODE, and then connecting everything into it*
  d. *You can have anything you want in the header*
  e. *Let's try the other one and see what happens*
ii. *Making* `amVK_SurfacePresenter` *Code part*
  a. *EveryBody is gonna depend on this*
  b. *They are only gonna keep a pointer to this parent*
  c. *from this one, they are gonna get everything that they need*

d. *even the VkDevice*
e. *It's like having all the nodes inside a TOP LEVEL FRAME NODE*
f. *Separating Code into .hh & .cpp is kinda crazy..... You basically can't have anything in the header....*
g. *i already tried this*

🎬 So far, The result [🔗 GITHUB]

- 📂 `common`
  - ◦ 📝 *amVK.hh*
  - ◦ 📝 *amVK_ColorSpace.hh*
  - ◦ 📝 *amVK_Enum2String.cpp*
  - ◦ 📝 *amVK_Enum2String.hh*
  - ◦ 📝 *amVK_GPU.hh*
  - ◦ 📝 *amVK_RenderPass_Descriptors.hh*
  - ◦ 📝 *amVK_log.cpp*
  - ◦ 📝 *amVK_log.hh*
- 📂 `core`
  - ◦ 📝 *amVK_Instance.hh*
  - ◦ 📝 *amVK_Device.hh*
  - ◦ 📝 *amVK_DeviceQCI.hh*
  - ◦ 📝 *amVK_Surface.hh*
  - ◦ 📝 *amVK_SwapChain.hh*
  - ◦ 📝 *amVK_SwapChainIMGs.hh*
  - ◦ 📝 *amVK_RenderPass.hh*
  - ◦ 📝 *amVK_RenderPassFBs.hh*
  - ◦ 📝 *amVK_CommandPool.hh*

- 📝 ***amVK_SurfacePresenter.hh***

- 📂 `extras`
  - ◦ 📝 *SCREENSHOT_STUDIO.hh*
  - ◦ 📝 *amVK_CommandBuffer.hh*
  - ◦ 📝 *amVK_FrameBuffer.hh*
  - ◦ 📝 *amVK_Image.hh*
  - ◦ 📝 *amVK_SemaPhone.hh*
- 📂 `guide`
  - ◦ *(Directory placeholder – add guide files here if any)*
- 📂 `impl`
  - ◦ 📝 *amVK_Device.cpp*
  - ◦ 📝 *amVK_Instance.cpp*
  - ◦ 📝 *amVK_InstanceProps.cpp*
  - ◦ 📝 *amVK_InstancePropsExport.cpp*
  - ◦ 📝 *amVK_InstancePropsExport_nloh...*
  - ◦ 📝 *amVK_Surface.cpp*
  - ◦ 📝 *amVK_SurfacePresenter.cpp*
  - ◦ 📝 *amVK_SwapChain.cpp*

10. `Windows WndProc`

SCREENSHOT1

Feature	WM_PAINT	WM_PRINT
Purpose	Sent by the system to request that a window redraw its client area.	Sent by an application to request that a window draw itself into a specified device context (e.g., for printing or capturing).
Trigger	Automatically triggered by the system when the client area becomes invalid (e.g., resizing, minimizing).	Explicitly sent by an application using SendMessage to request the window to draw itself.
Message ID	0x800F	0x0317
Who Sends It	Sent by the system.	Sent by the application (e.g., using SendMessage(hwnd, WM_PRINT, ...)).
Default Behavior	Calls the window's WndProc to handle the redraw.	Calls the window's WndProc to handle the drawing into the specified device context.
Device Context	Uses the device context provided by BeginPaint and EndPaint.	Uses the device context passed in wParam.
Use Case	Used for normal window redrawing (e.g., after invalidation or resizing).	Used for off-screen rendering, printing, or capturing the window's content.
System-Generated	Yes, automatically generated when the client area is invalid.	No, must be explicitly sent by the application.
Parameters	- `wParam` : Not used. - `lParam` : Not used.	- `wParam` : Handle to the device context (HDC). - `lParam` : Flags specifying what to draw.
Flags in lParam	Not applicable.	Flags include: - `PRF_CHECKVISIBLE` : Only draw if the window is visible. - `PRF_CHILDREN` : Draw child windows. - `PRF_CLIENT` : Draw the client area. - `PRF_NONCLIENT` : Draw the non-client area. - `PRF_ERASEBKGND` : Erase the background.
Child Windows	Does not automatically draw child windows.	Can optionally draw child windows if the PRF_CHILDREN flag is set.
Non-Client Area	Does not draw the non-client area (e.g., title bar, borders).	Can optionally draw the non-client area if the PRF_NONCLIENT flag is set.
Example Usage	Used in the WndProc to handle normal window painting.	Used for capturing the window's content into a bitmap or for printing.

Screenshot2

alt text

alt text

classDiagram class WM_PAINT { <<Message>> Purpose: "Request window redraw (client area)" Trigger: "System (automatic)" MessageID: "0x800F" DefaultBehavior: "Calls WndProc" DeviceContext: "BeginPaint/EndPaint" UseCase: "Normal window redrawing" Parameters: "wParam/lParam unused" ChildWindows: "No" NonClientArea: "No" } class WM_PRINT { <<Message>> Purpose: "Request window draw into HDC" Trigger: "Application (manual)" MessageID: "0x0317" DefaultBehavior: "Calls

WndProc" DeviceContext: "HDC in wParam" UseCase: "Printing/capturing" Parameters: "wParam: HDC \n lParam: Flags \n (PRF_CHECKVISIBLE, PRF_CHILDREN...)" ChildWindows: "Optional (PRF_CHILDREN)" NonClientArea: "Optional (PRF_NONCLIENT)" } WM_PAINT --|> SystemGenerated WM_PRINT --|> ApplicationGenerated