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. Notes on RenderLoop

- ***There are a few ways that you can record*** `CMDBUF`
  1. *Recording a* `VkCommandBuffer` *only Once*
  2. *Recording a* `VkCommandBuffer` *more than Once*
     - *more than Once --> requires,* `CMDBUF` *to be reset by* `vkResetCommandBuffer()` *before recording again*
       - *Note:-* `vkBeginCommandBuffer()` *also does do an implicit reset*
       - *which I don't believe should exist* 🧑‍🏫
       - *I always believe "Explicit is better than implicit"*
     - *REQ:-*
       - `VkCommandPoolCreateInfo.flags` ▨ `VK_COMMAND_POOL_CREATE_RESET_COMMAND_BUFFER_BIT`
  3. *Submitting a* `VkCommandBuffer` *only Once*
     - *REQ:-*
       - `VkCommandBufferBeginInfo.flags` ▨ `VK_COMMAND_BUFFER_USAGE_ONE_TIME_SUBMIT_BIT`
     - *People usually uses* `SUBMIT_ONCE` *with a* `RESET_CMDBUF`
  4. *Implementing Synchronization features is so fked up. After Present Image, call* `vkQueueWaitIdle()`

3. Enabling 🛡 Validation Layers 🍰

```
class amVK_InstanceProps {
  public:
    static inline           REY_Array<VkLayerProperties>              amVK_1D_InstanceLayers;
    #define amVK_LOOP_ILayers(_var_) for (uint32_t _var_ = 0,  lim = amVK_1D_InstanceLayers.n;   _var_ < lim;  _var_++)

    static inline bool called_EnumerateInstanceLayerProperties = false;
    static void                EnumerateInstanceLayerProperties(void);                 // amVK_1D_InstanceLayers

    static bool                       isInstanceLayerAvailable(const char *layerName); // amVK_1D_InstanceLayers
}

class amVK_Instance {
    static inline REY_ArrayDYN<char*> amVK_1D_Instance_Layers_Enabled;
    static void                       addTo_1D_Instance_Layers_Enabled(const char* layerName);
    static void                         log_1D_Instance_Layers_Enabled(VkResult ret);  // CreateDevice() calls this
}

amVK_Instance::addTo_1D_Instance_Layers_Enabled("VK_LAYER_KHRONOS_validation");
```

🔗 `GITHUB`

amVK_Instance.hh#L25

amVK_InstanceProps.hh#L13

amVK_InstanceProps.cpp#L66

4. Coloring Window

5. Semaphore & Synchronization

6. VkQueueWaitIdle

4. `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` 📝

5. Command Recording

1. `VkCommandBufferBeginInfo`
    - https://vkdoc.net/man/VkCommandBufferBeginInfo
        - `.sType` ▨ `VK_STRUCTURE_TYPE_COMMAND_BUFFER_BEGIN_INFO`
        - `.pNext` ▨ `nullptr`
            - 🪐 `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` 🪐 `nullptr`
        - `.renderPass` ▨ 🧑‍💻
        - `.framebuffer` ▨ 🧑‍💻
        - `.renderArea`
            - *https://vkdoc.net/man/VkRect2D*
        - `.pClearValues`
            - *https://vkdoc.net/man/VkClearValue*

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*

Before Commit:- *https://github.com/REYNEP/amGHOST/blob/9cec3e58db123144bd8d88363ccf9a4a7ffc9edc/amVK/amVK_Surface.hh*
Middle (Discarded) Commit:- *https://github.com/REYNEP/amGHOST/blob/3be7cfcd154b383cd98783d302468f63fda0618b/amVK/amVK_SurfacePresenter.hh*
Final Commit:- *https://github.com/REYNEP/amGHOST/blob/7376cdb5c2c6eee19655dae436e6cf8edd02e1d5/amVK/amVK_SurfacePresenter.hh*

`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*

7. `Back 2 Command Recording`

3. `vkBeginCommandBuffer()`

- https://vkdoc.net/man/vkBeginCommandBuffer
  - `.commandBuffer` ▨ 🧑‍💻
  - `.pBeginInfo` ▨ 🧑‍💻
- `</> TheCode`

```cpp
amVK_CommandPool {
  public:
    REY_Array<VkCommandBuffer>       vk_CommandBuffers;
    REY_Array<VkCommandBuffer> AllocateCommandBuffers(void);

  public:
    VkCommandBufferBeginInfo BI = {
        .sType = VK_STRUCTURE_TYPE_COMMAND_BUFFER_BEGIN_INFO,
        .pNext = 0,
        .flags = VK_COMMAND_BUFFER_USAGE_ONE_TIME_SUBMIT_BIT,
        .pInheritanceInfo = nullptr
    };
    void BeginCommandBuffer(uint32_t CMDBUF_Index) {
        VkResult return_code = vkBeginCommandBuffer(vk_CommandBuffers[CMDBUF_Index], &BI);
        amVK_return_code_log( "vkBeginCommandBuffer()" );
    }
}
```

4. `vkCmdBeginRenderPass()`

- https://vkdoc.net/man/vkCmdBeginRenderPass
  - `.commandBuffer` ▨ 🧑‍💻
  - `.pRenderPassBegin` ▨ 🧑‍💻
  - `.contents` 🔠 `VK_SUBPASS_CONTENTS_INLINE`
    - *https://vkdoc.net/man/VkSubpassContents* | *ivirtex-github*
      - 🔠 `INLINE`
      - 🔠 `SECONDARY_COMMAND_BUFFERS` *[secondary command buffer]*
      - 🔠 `INLINE_AND_SECONDARY_COMMAND_BUFFERS_KHR` *[VK_KHR_maintenance7]*
      - 🔠 `INLINE_AND_SECONDARY_COMMAND_BUFFERS_EXT` *[VK_EXT_nested_command_buffer]*

5. `vkCmdSetViewport()`

- https://vkdoc.net/man/vkCmdSetViewport
  - `.commandBuffer` ▨ 🧑‍💻
  - `.firstViewport` ▨ `0`
  - `.viewportCount` ▨ `1`
  - `.pViewports` ▨ `VkViewport`
    - *https://vkdoc.net/man/VkViewport*

6. `vkCmdSetScissor()`

- https://vkdoc.net/man/vkCmdSetScissor
  - `.pScissors` ▨ `VkRect2D`
    - *https://vkdoc.net/man/VkRect2D*

7. `vkCmdEndRenderPass()`

- https://vkdoc.net/man/vkCmdEndRenderPass
  - `.commandBuffer` ▨ 🧑‍💻

8. `vkEndCommandBuffer()`

- https://vkdoc.net/man/vkEndCommandBuffer
  - `.commandBuffer` ▨ 🧑‍💻

8. `Submit Command Buffer`

1. `VkSubmitInfo`
   - https://vkdoc.net/man/VkSubmitInfo
     - `.sType` ▨ `VK_STRUCTURE_TYPE_SUBMIT_INFO`
     - `.pNext` ⊘ `NULL`
     - `.pWaitSemaphores` 🔗 *Chapter9.1*
       - ▨ `amVK_SwapChain::AcquireNextImage_SemaPhore`
     - `.pWaitDstStageMask` ▨ `VK_PIPELINE_STAGE_COLOR_ATTACHMENT_OUTPUT_BIT`
     - `.pCommandBuffers` ▨ 🧑‍💻
     - `.pSignalSemaphores`
       - ▨ `amVK_SurfacePresenter::RenderingFinished_SemaPhore`

---

2. `vkQueueSubmit()`
   - https://vkdoc.net/man/vkQueueSubmit
     - `.queue` ▨ `GraphicsQueue`
     - `.submitCount` ▨ `1`
     - `.pSubmits` ▨ 🧑‍💻
     - `.fench` ▨ `VK_NULL_HANDLE`
3. `vkGetDeviceQueue()`
   - https://vkdoc.net/man/vkGetDeviceQueue
     - `.device`
     - `.queueFamilyindex` 🔗 *Chapter2.7*
       - `amVK_Device::amVK_1D_QCIs::select_QFAM_Graphics()`
     - `.queueIndex` 🔗 *Chapter2.4*
       - `VkDeviceQueueCreateInfo.queueCount`
     - `.pQueue` ↩ 📦

---

4. `VkPresentInfoKHR`
   - https://vkdoc.net/man/VkPresentInfoKHR
     - `.sType` ▨ `VK_STRUCTURE_TYPE_PRESENT_INFO_KHR`
     - `.pNext` ⊘ `NULL`
       - ⊘ *Maybe some interesting extensions, idk*
     - `.pWaitSemaphores` 🔗 *Chapter9.6*
       - ▨ `amVK_SwapChain::RenderingFinished_SemaPhore`
     - `.pSwapchains` ▨ 🧑‍💻
     - `.pImageIndices`
     - `.pResults`
5. `vkQueuePresentKHR()`
   - https://vkdoc.net/man/vkQueuePresentKHR
     - `.queue` ▨ 🧑‍💻
     - `.pPresentInfo` ▨ 🧑‍💻
6. `📽 So far, The result` *[*🔗 `GITHUB` *]*
   - https://github.com/REYNEP/amGHOST/tree/805df077be97835083dd7716c3597b2f0e9347cb/amVK

9. `RenderLoop`

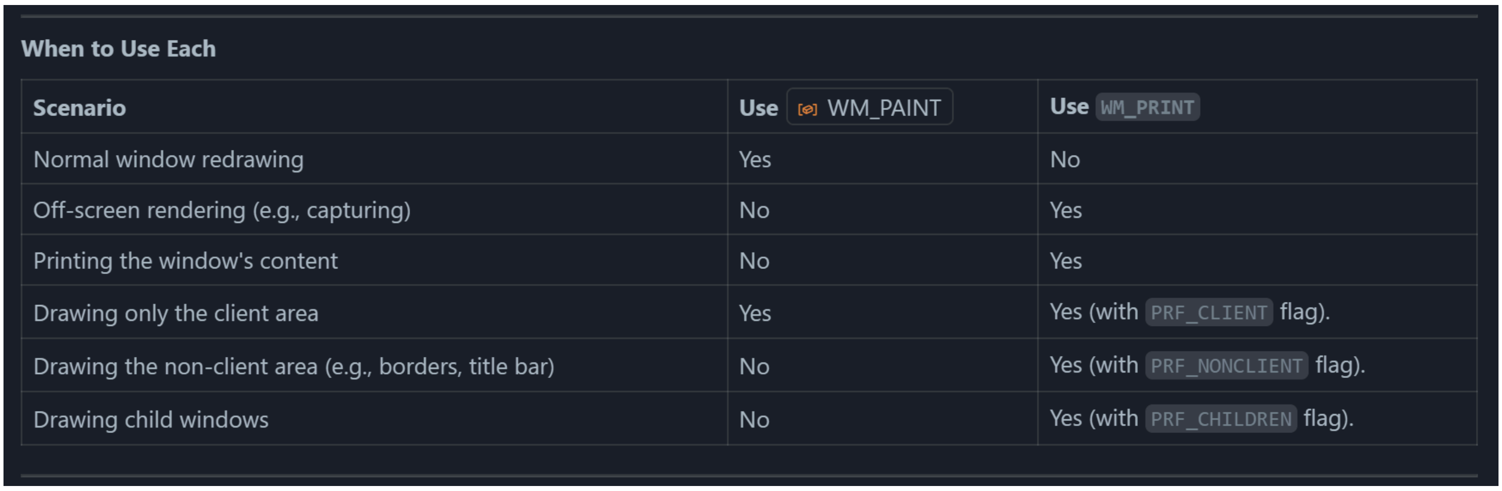
1. ***amGHOST Events <-- (Win32/XCB/X11/Wayland/macOS)***
   1. *EventTypes*
      - https://www.youtube.com/watch?v=xnopUoZbMEk&list=PLlrATfBNZ98dC-V-N3m0Go4deliWHPFwT&index=9
2. ***EventLoop***
3. ***Window ReSizing + (SwapChain ReCreation)***
4. ***REY_NoobTimer::wait(10);***
5. ***On windows, if you don't ask win32 for the Events of the WINDOW, then windows is gonna go insane, it's gonna flag your app as "Not Reposnding"***

10. `Windows WndProc`

SCREENSHOT1

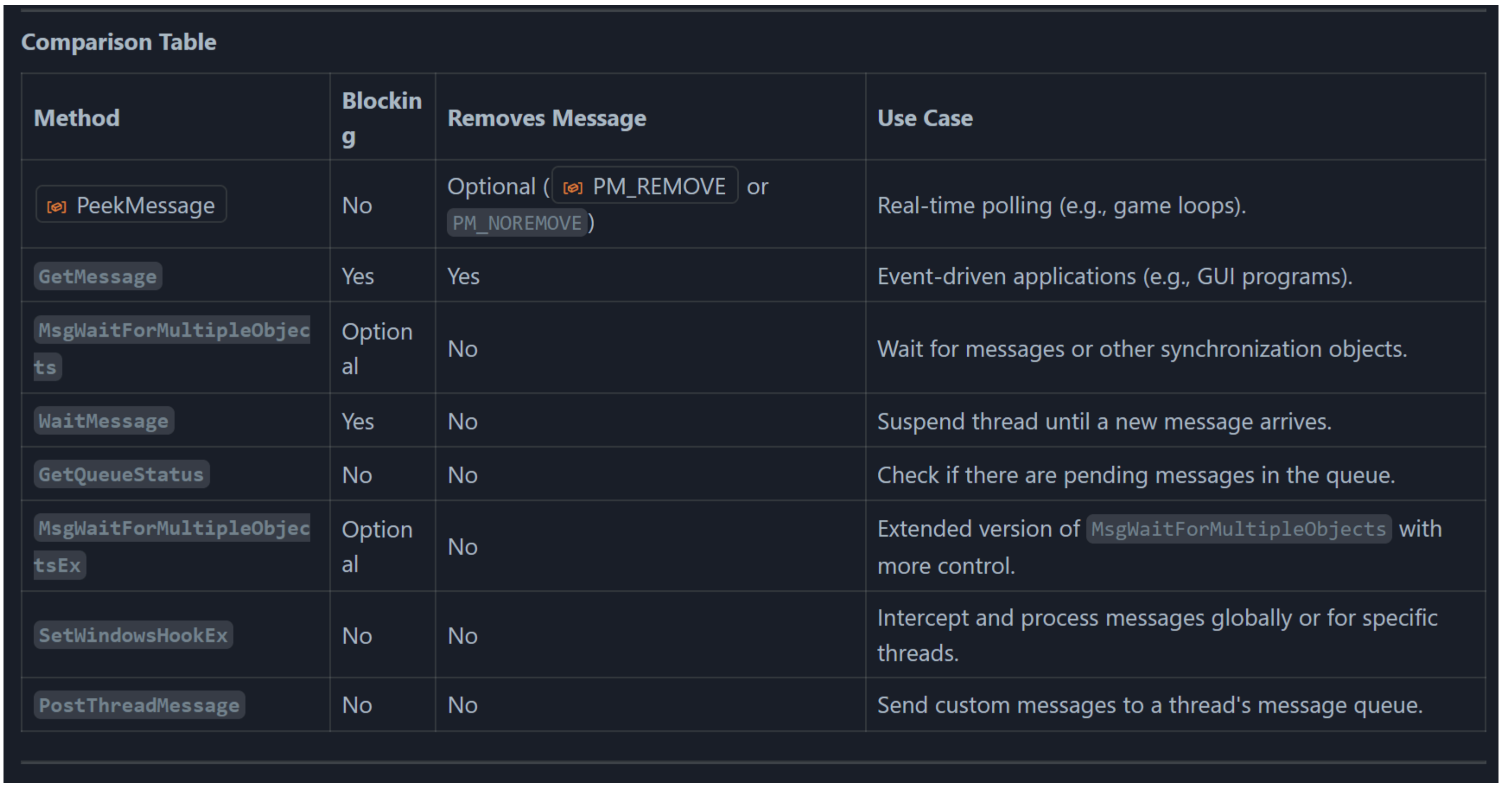
Feature	WM_PAINT	WM_PRINT
Purpose	Sent by the system to request that a window redraw its client area.	Sent by an application to request that a window draw itself into a specified device context (e.g., for printing or capturing).
Trigger	Automatically triggered by the system when the client area becomes 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.

Feature	WM_PAINT	WM_PRINT
		- `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.



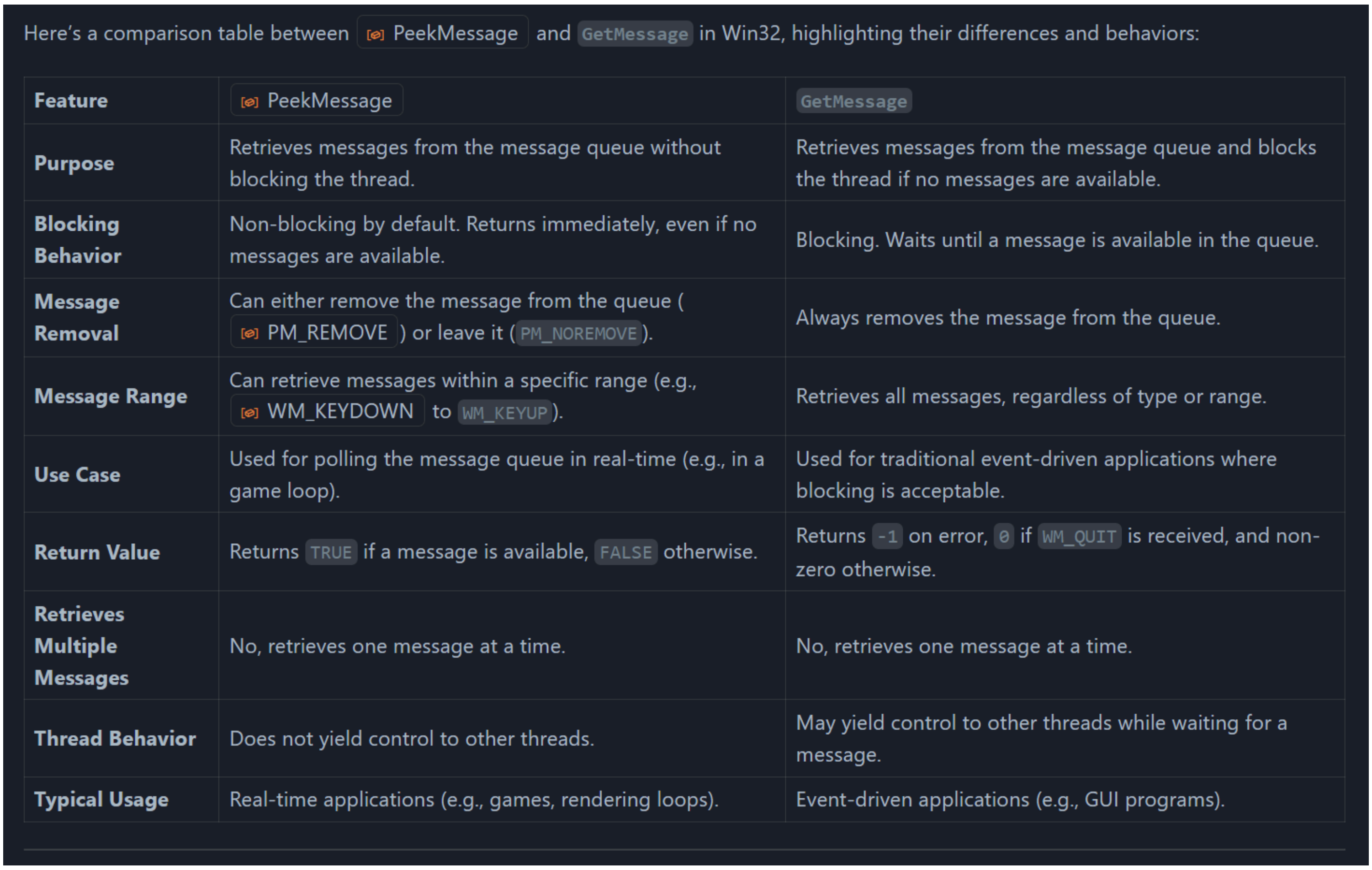
When to Use Each

Scenario	Use `WM_PAINT`	Use `WM_PRINT`
Normal window redrawing	Yes	No
Off-screen rendering (e.g., capturing)	No	Yes
Printing the window's content	No	Yes
Drawing only the client area	Yes	Yes (with `PRF_CLIENT` flag).
Drawing the non-client area (e.g., borders, title bar)	No	Yes (with `PRF_NONCLIENT` flag).
Drawing child windows	No	Yes (with `PRF_CHILDREN` flag).



Comparison Table

Method	Blocking	Removes Message	Use Case
`PeekMessage`	No	Optional (`PM_REMOVE` or `PM_NOREMOVE`)	Real-time polling (e.g., game loops).
`GetMessage`	Yes	Yes	Event-driven applications (e.g., GUI programs).
`MsgWaitForMultipleObjects`	Optional	No	Wait for messages or other synchronization objects.
`WaitMessage`	Yes	No	Suspend thread until a new message arrives.
`GetQueueStatus`	No	No	Check if there are pending messages in the queue.
`MsgWaitForMultipleObjectsEx`	Optional	No	Extended version of `MsgWaitForMultipleObjects` with more control.
`SetWindowsHookEx`	No	No	Intercept and process messages globally or for specific threads.
`PostThreadMessage`	No	No	Send custom messages to a thread's message queue.



Here's a comparison table between PeekMessage and `GetMessage` in Win32, highlighting their differences and behaviors:

Feature	PeekMessage	`GetMessage`
Purpose	Retrieves messages from the message queue without blocking the thread.	Retrieves messages from the message queue and blocks the thread if no messages are available.
Blocking Behavior	Non-blocking by default. Returns immediately, even if no messages are available.	Blocking. Waits until a message is available in the queue.
Message Removal	Can either remove the message from the queue (PM_REMOVE) or leave it (`PM_NOREMOVE`).	Always removes the message from the queue.
Message Range	Can retrieve messages within a specific range (e.g., WM_KEYDOWN to `WM_KEYUP`).	Retrieves all messages, regardless of type or range.
Use Case	Used for polling the message queue in real-time (e.g., in a game loop).	Used for traditional event-driven applications where blocking is acceptable.
Return Value	Returns `TRUE` if a message is available, `FALSE` otherwise.	Returns `-1` on error, `0` if `WM_QUIT` is received, and non-zero otherwise.
Retrieves Multiple Messages	No, retrieves one message at a time.	No, retrieves one message at a time.
Thread Behavior	Does not yield control to other threads.	May yield control to other threads while waiting for a message.
Typical Usage	Real-time applications (e.g., games, rendering loops).	Event-driven applications (e.g., GUI programs).