Introduction to Computer Graphics

186.832, 2021W, 3.0 ECTS

Vulkan Lecture Series, Episode 2:

Swap Chain

Johannes Unterguggenberger

Institute of Visual Computing & Human-Centered Technology

TU Wien, Austria

The Swap Chain

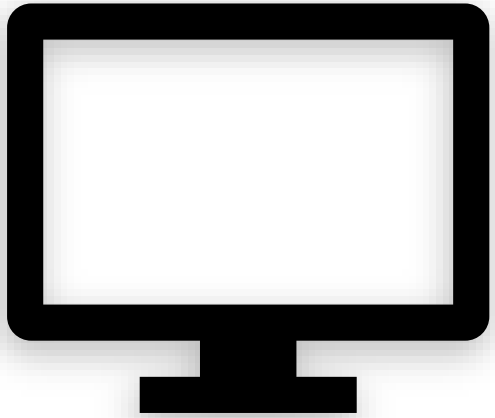
Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

Swap Chain

available images:

presented image:



The Swap Chain

Application/Render Loop

```
while (true) {

  acquireNextImage();

  draw();

  present();

}
```

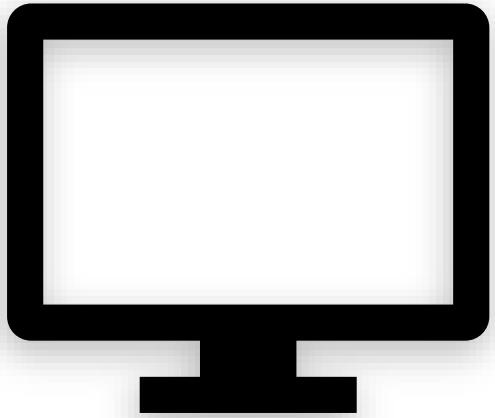
Image 1

Swap Chain

available images:

Image 2

presented image:



The Swap Chain

Application/Render Loop

```
while (true) {

  acquireNextImage();

  draw();

  present();

}
```

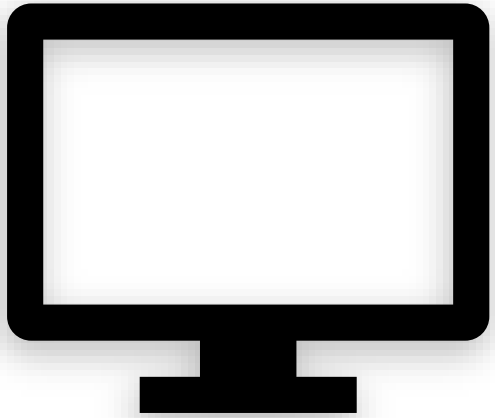
Image 1

Swap Chain

available images:

Image 2

presented image:



The Swap Chain

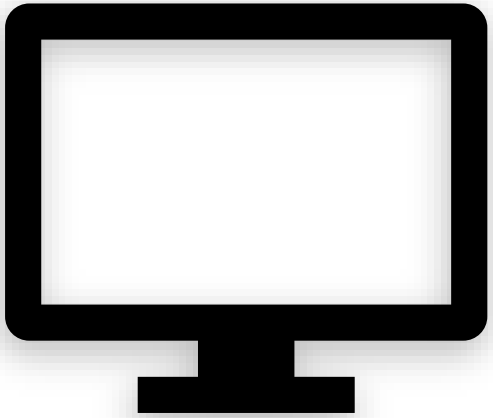
Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

Swap Chain

available images:

presented image:



The Swap Chain

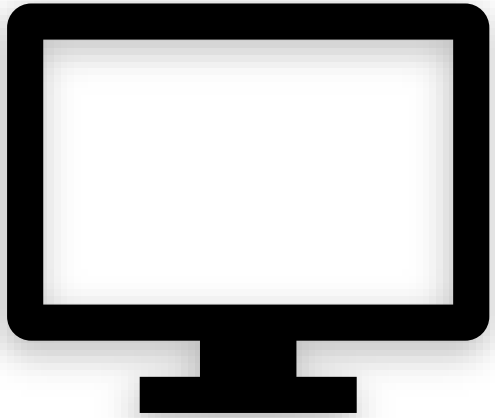
Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

Swap Chain

available images:

presented image:



The Swap Chain

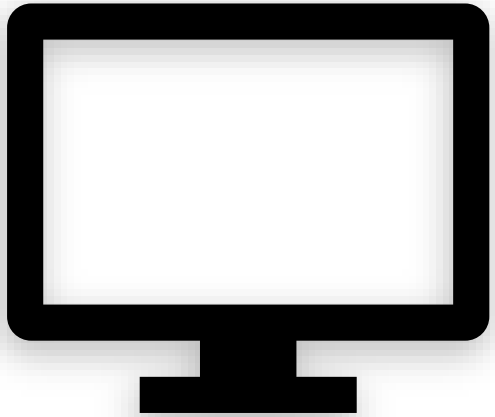
Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

Swap Chain

available images:

presented image:



The Swap Chain

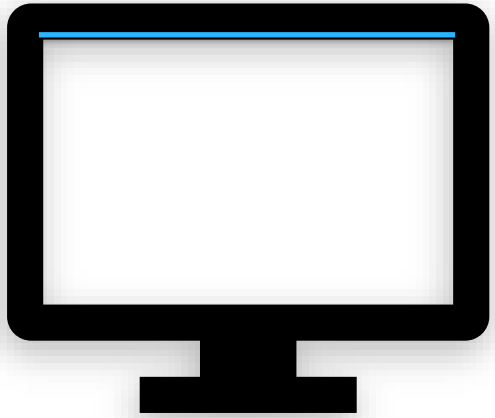
Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

Swap Chain

available images:

presented image:



The Swap Chain

Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

Swap Chain

available images:

presented image:

The Swap Chain

Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

Swap Chain

available images:

presented image:

The Swap Chain

Application/Render Loop

```
while (true) {

  acquireNextImage();

  draw();

  present();

}
```

Image 2

Swap Chain

available images:

presented image:

The Swap Chain

Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

Swap Chain

available images:

presented image:

The Swap Chain

Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

Swap Chain

available images:

presented image:

The Swap Chain

Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

Swap Chain

available images:

presented image:

The Swap Chain

Application/Render Loop

```
while (true) {

  acquireNextImage();

  draw();

  present();

}
```

Swap Chain

available images:

presented image:

The Swap Chain

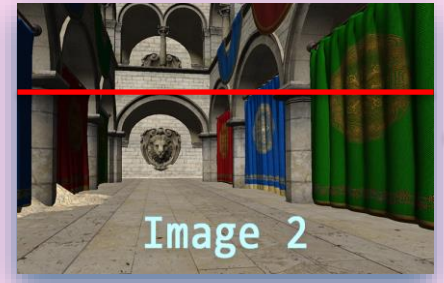
Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

Swap Chain

available images:

presented image:



The Swap Chain

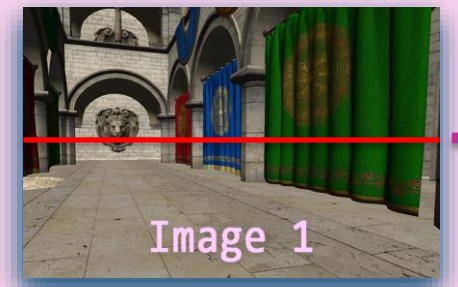
Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

Swap Chain

available images:

presented image:



The Swap Chain

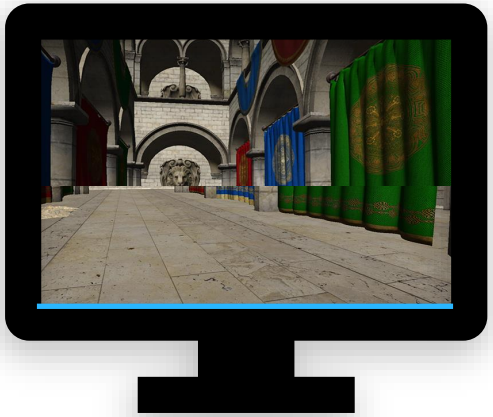
Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

Swap Chain

available images:

presented image:



The Swap Chain

Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

Swap Chain

available images:

presented image:

The Swap Chain

Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

Swap Chain

available images:

presented image:

The Swap Chain

Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

Swap Chain

available images:

presented image:

The Swap Chain

Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

Swap Chain

available images:

presented image:

"Immediate"

Presentation Mode

The Swap Chain

Application/Render Loop

```
while (true) {

  acquireNextImage();

  draw();

  present();

}
```

Swap Chain

available images:

presented image:

"Immediate"

Presentation Mode

The Swap Chain

Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

Swap Chain

available images:

presented image:

"Immediate"

Presentation Mode

The Swap Chain

Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

Swap Chain

available images:

presented image:

"Immediate"

Presentation Mode

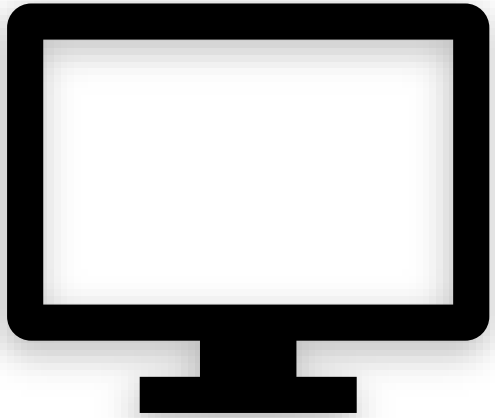
Presentation Modes:

"Immediate"
Presentation Mode

"FIFO"
Presentation Mode

"FIFO Relaxed"
Presentation Mode

"Mailbox"
Presentation Mode



Presentation Modes:

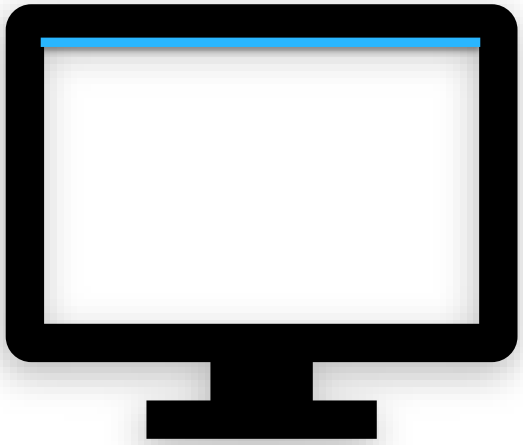
"Immediate"
Presentation Mode

"FIFO"
Presentation Mode

"FIFO Relaxed"
Presentation Mode

"Mailbox"
Presentation Mode

e.g.:
60 Hz monitor



Presentation Modes:

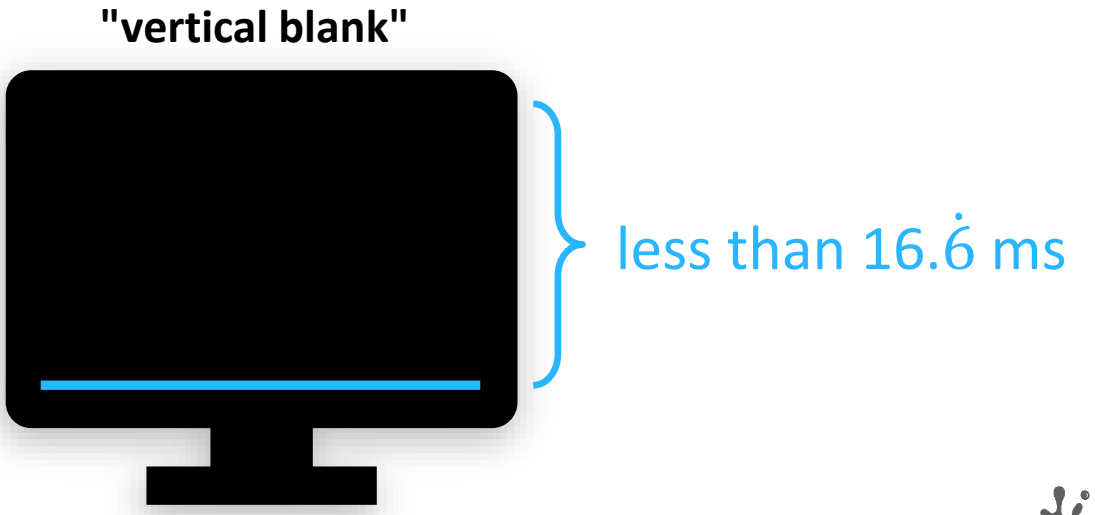
"Immediate"
Presentation Mode

"FIFO"
Presentation Mode

"FIFO Relaxed"
Presentation Mode

"Mailbox"
Presentation Mode

e.g.:
60 Hz monitor



Vertical Blank

Presentation Modes:

"Immediate"
Presentation Mode

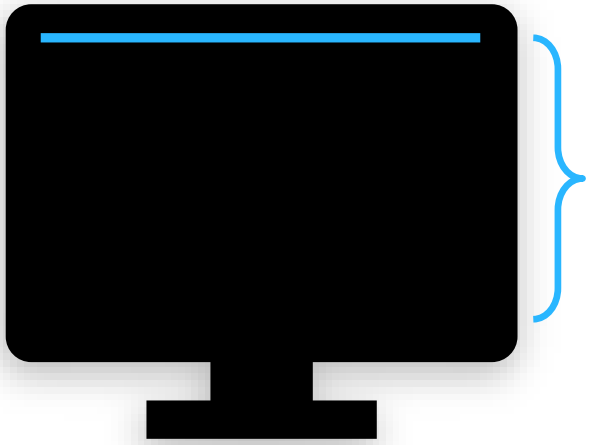
"FIFO"
Presentation Mode

"FIFO Relaxed"
Presentation Mode

"Mailbox"
Presentation Mode

e.g.:
60 Hz monitor

"vertical blanking period"



less than $16.\dot{6}$ ms

Vertical Blank

Presentation Modes:

"Immediate"
Presentation Mode

"FIFO"
Presentation Mode

"FIFO Relaxed"
Presentation Mode

"Mailbox"
Presentation Mode

e.g.:
60 Hz monitor

Vertical Blank

Presentation Modes:

"Immediate"
Presentation Mode

"FIFO"
Presentation Mode

"FIFO Relaxed"
Presentation Mode

"Mailbox"
Presentation Mode

e.g.:
144 Hz monitor

Vertical Blank

Presentation Modes:

"Immediate"
Presentation Mode

"FIFO"
Presentation Mode

"FIFO Relaxed"
Presentation Mode

"Mailbox"
Presentation Mode

e.g.:
30 - 144 Hz monitor
(adaptive sync)

Presentation Modes:

"Immediate"
Presentation Mode

"FIFO"
Presentation Mode

"FIFO Relaxed"
Presentation Mode

"Mailbox"
Presentation Mode

does not wait

does wait

waits, unless frame is running late

does wait

"vertical blank"

Immediate

Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

Swap Chain

available images:

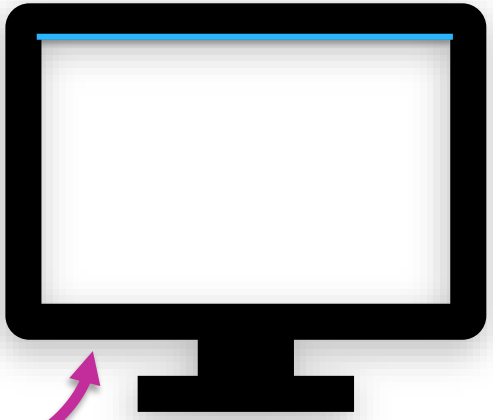
to be presented:

presented image:

"Immediate"

Presentation Mode

"vertical blank"



Immediate

Application/Render Loop

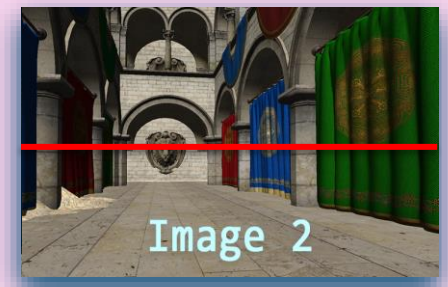
```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

Swap Chain

available images:

to be presented:

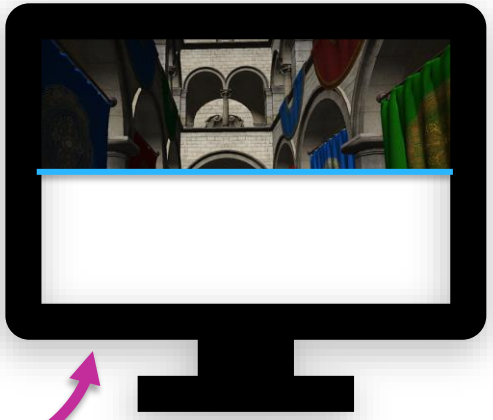
presented image:



"Immediate"

Presentation Mode

"vertical blank"



Immediate

Application/Render Loop

```
while (true) {

  acqu

  draw

  present();

}
```

Swap Chain

available images:

presented image:

"Immediate"

Presentation Mode

VK_PRESENT_MODE_IMMEDIATE_KHR specifies that the presentation engine does not wait for a vertical blanking period to update the current image, meaning this mode **may** result in visible tearing. No internal queuing of presentation requests is needed, as the requests are applied immediately.

The Khronos Group. Vulkan 1.2.196 Specificaton

FIFO

Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

Swap Chain

available images:

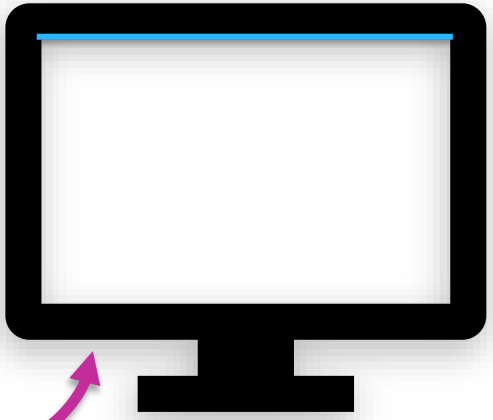
to be presented:

presented image:

"FIFO"

Presentation Mode

"vertical blank"



FIFO

Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

Swap Chain

available images:

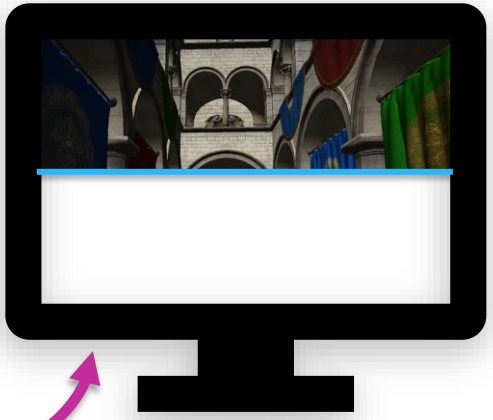
to be presented:

presented image:

"FIFO"

Presentation Mode

"vertical blank"



FIFO

Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

Swap Chain

available images:

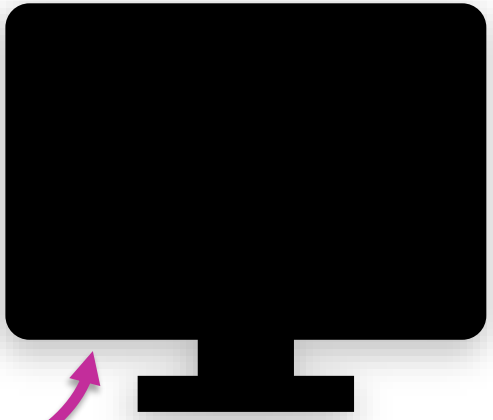
to be presented:

presented image:

"FIFO"

Presentation Mode

"vertical blank"



FIFO

Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

Swap Chain

available images:

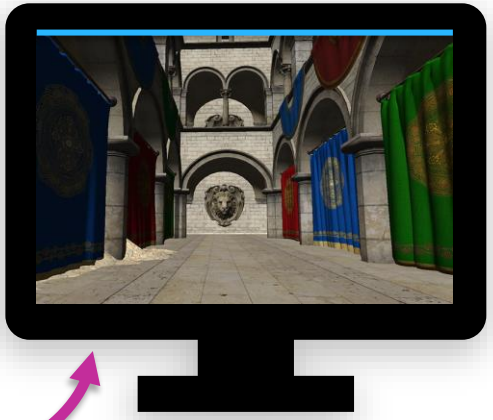
to be presented:

presented image:

"FIFO"

Presentation Mode

"vertical blank"



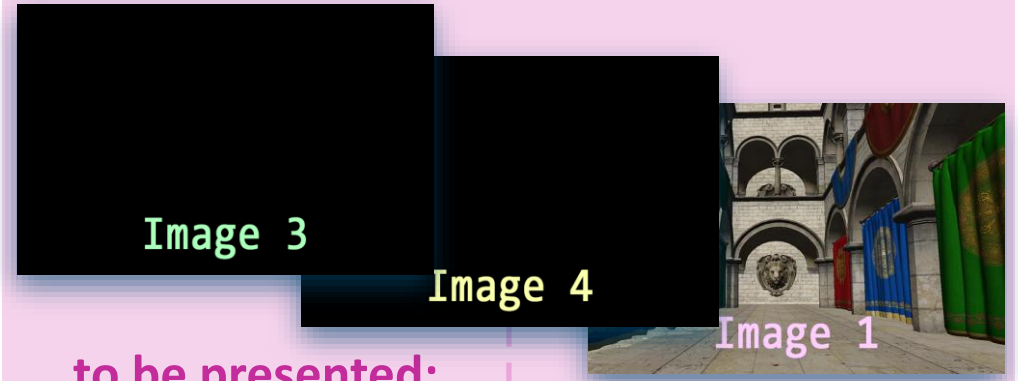
FIFO

Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

Swap Chain

available images:



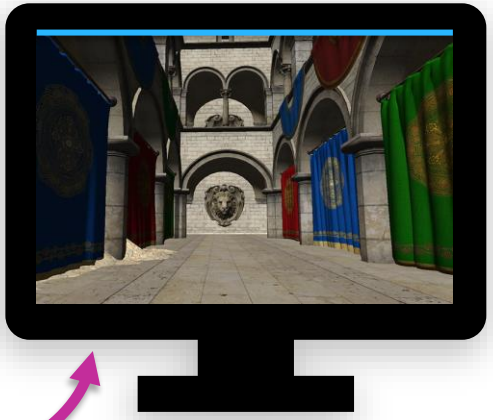
to be presented:

presented image:

"FIFO"

Presentation Mode

"vertical blank"



FIFO

Application/Render Loop

```
while (true) {

  acquireNextImage();

  draw();

  present();

}
```

Swap Chain

available images:

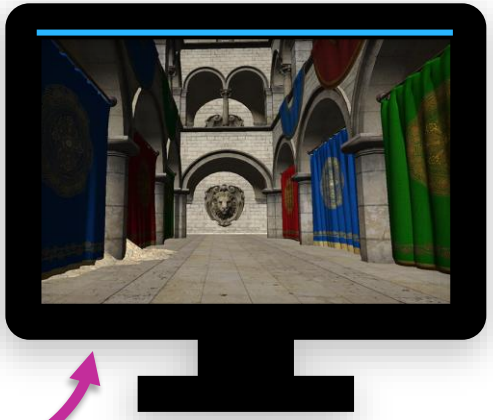
to be presented:

presented image:

"FIFO"

Presentation Mode

"vertical blank"



FIFO

Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

Swap Chain

available images:

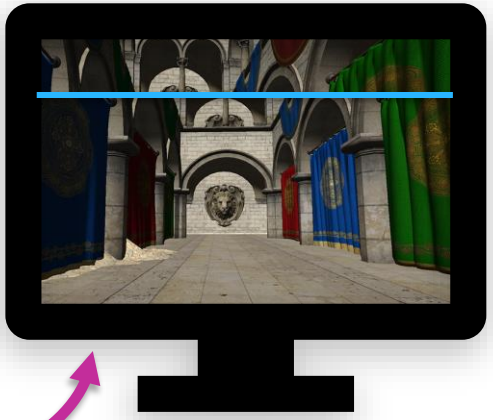
to be presented:

presented image:

"FIFO"

Presentation Mode

"vertical blank"



FIFO

Application/Render Loop

```
while (true) {

  acquireNextImage();

  draw();

  present();

}
```

Swap Chain

available images:

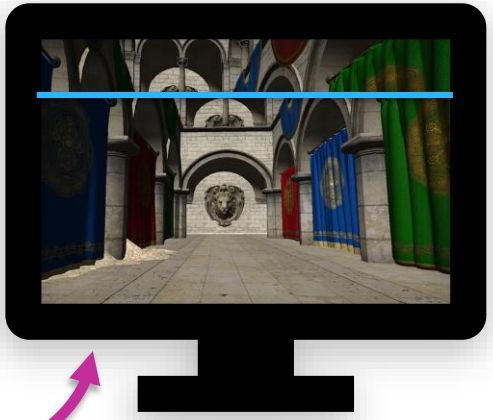
to be presented:

presented image:

"FIFO"

Presentation Mode

"vertical blank"



FIFO

Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

Swap Chain

available images:

to be presented:

presented image:

"FIFO"

Presentation Mode

"vertical blank"

FIFO

Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

Swap Chain

available images:

to be presented:

presented image:

"FIFO"

Presentation Mode

"vertical blank"

Application is running ahead

FIFO

Application/Render Loop

```
while (true) {

  acquireNextImage();


  draw();


  present();

}
```

Swap Chain

available images:

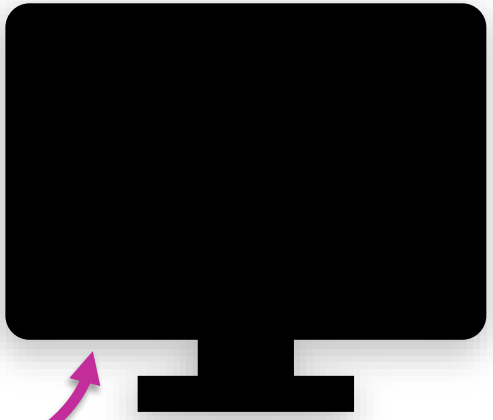
to be presented:

presented image:

"FIFO"

Presentation Mode

"vertical blank"



Application is running ahead

FIFO

Application/Render Loop

```
while (true) {

  acquireNextImage();


  draw();


  present();

}
```

Swap Chain

available images:

to be presented:

presented image:

"FIFO"

Presentation Mode

"vertical blank"

Application is running ahead

FIFO

Application/Render Loop

```
while (true) {

  acquireNextImage();


  draw();


  present();

}
```

Swap Chain

available images:

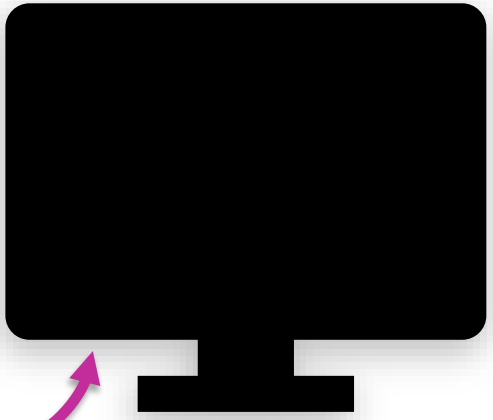
to be presented:

presented image:

"FIFO"

Presentation Mode

"vertical blank"



Application is running ahead

FIFO

Application/Render Loop

```
while (true) {

  acquireNextImage();


  draw();


  present();

}
```

Swap Chain

available images:

to be presented:

presented image:

"FIFO"

Presentation Mode

Application is running ahead

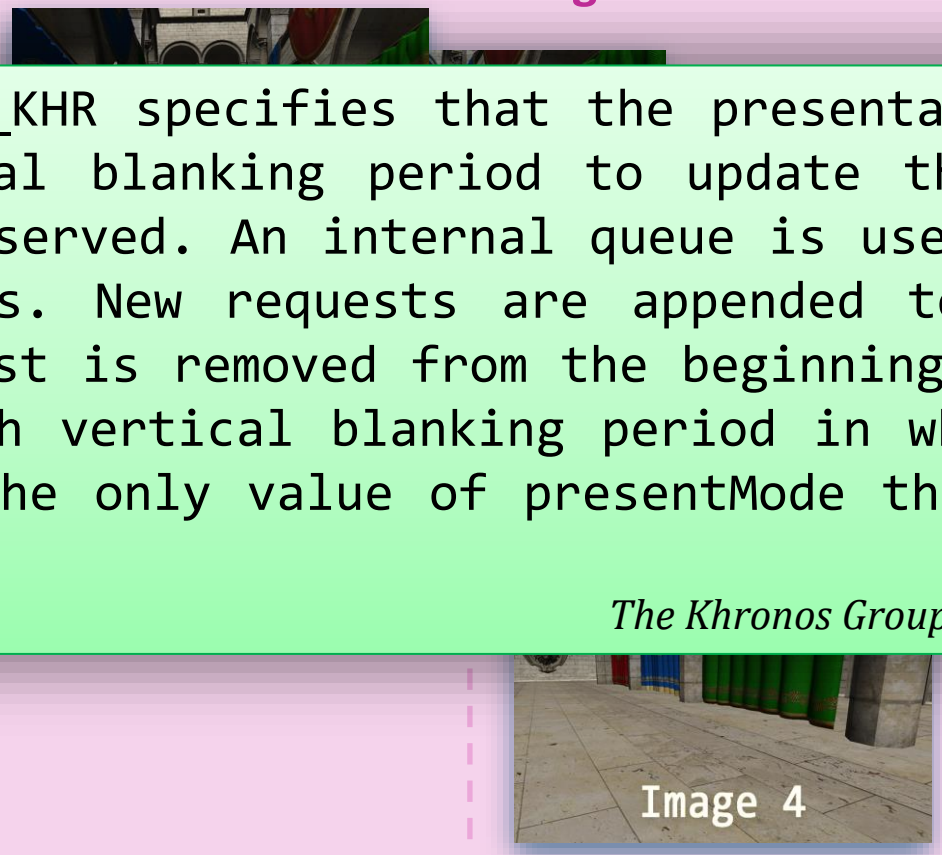
FIFO

Application/Render Loop

```
while (true) {

  acqu

  draw

  present();

}
```

Swap Chain

available images:



"FIFO"

Presentation Mode

VK_PRESENT_MODE_FIFO_KHR specifies that the presentation engine waits for the next vertical blanking period to update the current image. Tearing **cannot** be observed. An internal queue is used to hold pending presentation requests. New requests are appended to the end of the queue, and one request is removed from the beginning of the queue and processed during each vertical blanking period in which the queue is non-empty. This is the only value of presentMode that is **required** to be supported.

The Khronos Group. Vulkan 1.2.196 Specificaton

FIFO

Application/Render Loop

```
while (true) {

  acquireNextImage();

  draw();

  present();

}
```

Image 1

Swap Chain

available images:

Image 2

Image 3

Image 4

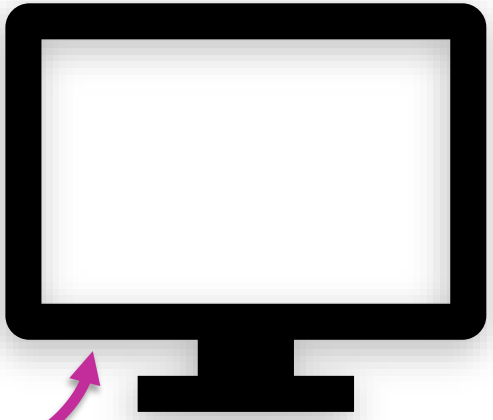
to be presented:

presented image:

"FIFO"

Presentation Mode

"vertical blank"



Application is lagging behind

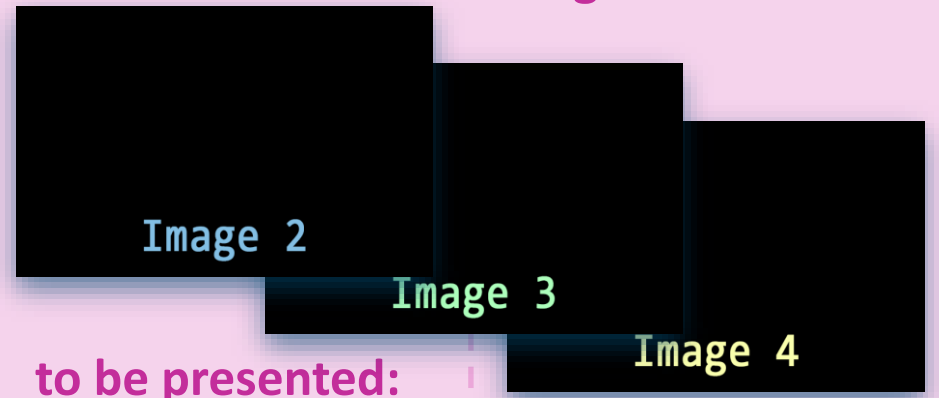
FIFO

Application/Render Loop

```
while (true) {

  acquireNextImage();

  draw();

  present();

}
```

Swap Chain

available images:



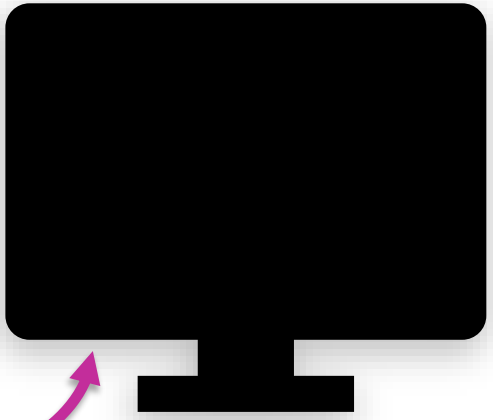
to be presented:

presented image:

"FIFO"

Presentation Mode

"vertical blank"



Application is lagging behind

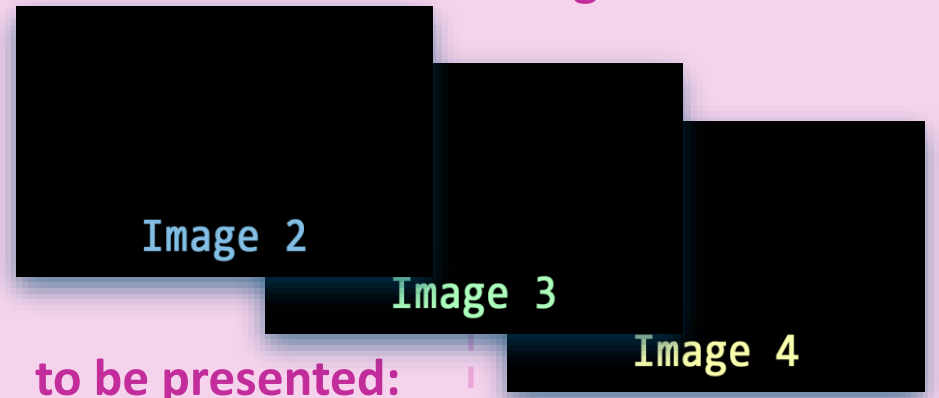
FIFO

Application/Render Loop

```
while (true) {

  acquireNextImage();


  draw();


  present();

}
```

Swap Chain

available images:



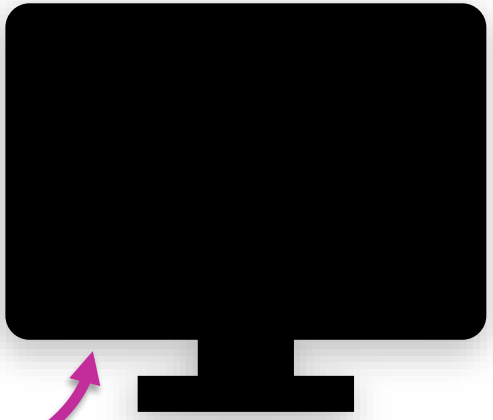
to be presented:

presented image:

"FIFO"

Presentation Mode

"vertical blank"



Application is lagging behind

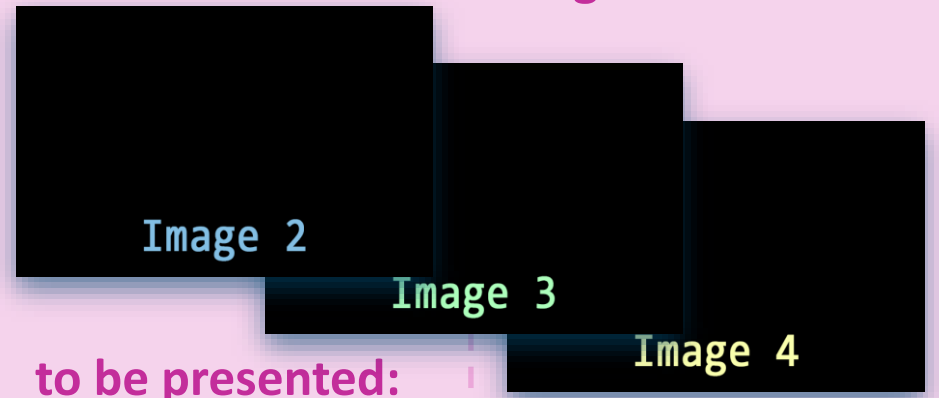
FIFO

Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();

  present();

}
```

Swap Chain

available images:



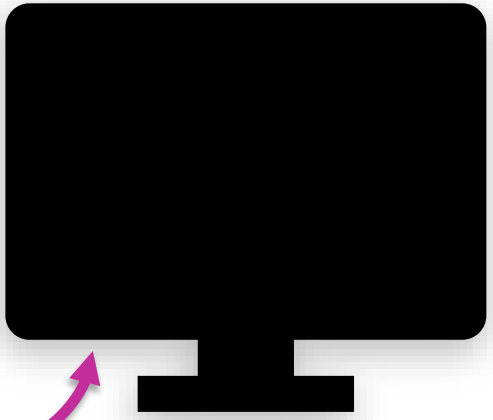
to be presented:

presented image:

"FIFO"

Presentation Mode

"vertical blank"



Application is lagging behind

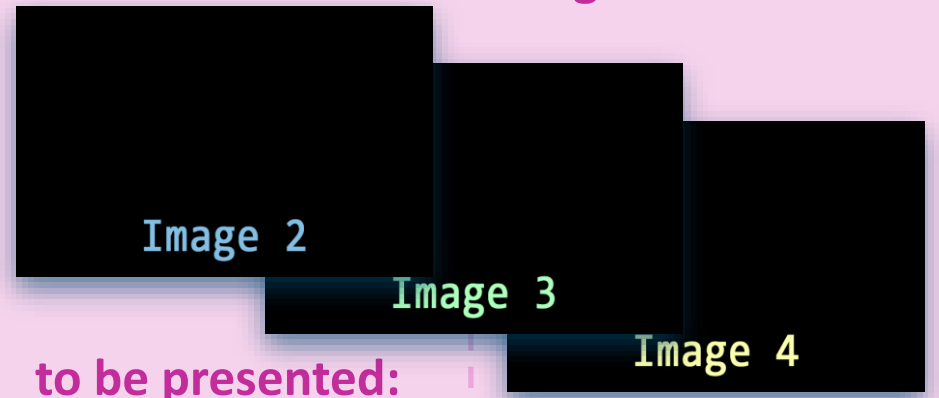
FIFO

Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

Swap Chain

available images:



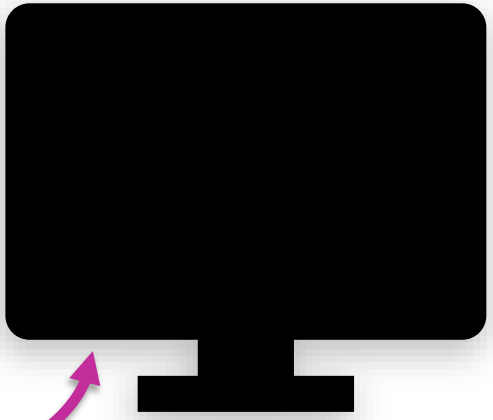
to be presented:

presented image:

"FIFO"

Presentation Mode

"vertical blank"



Application is lagging behind

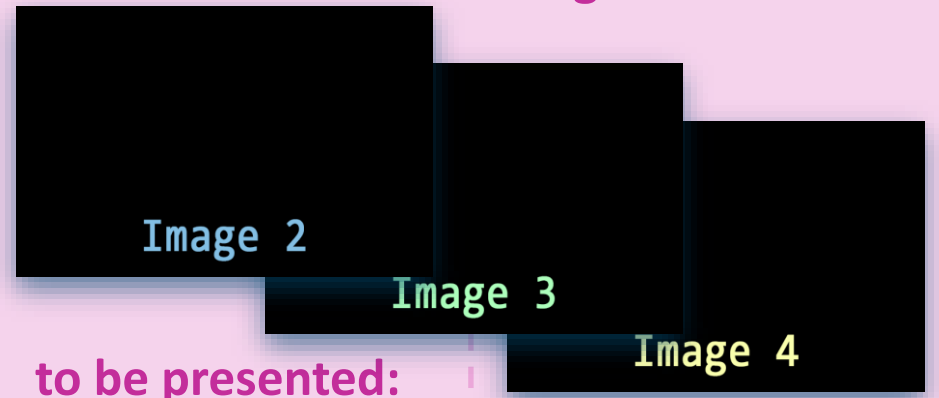
FIFO

Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

Swap Chain

available images:



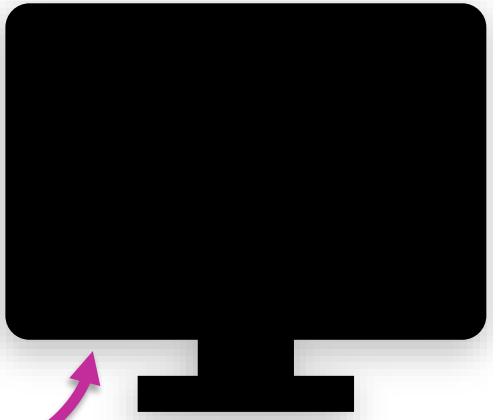
to be presented:

presented image:

"FIFO"

Presentation Mode

"vertical blank"



Application is lagging behind

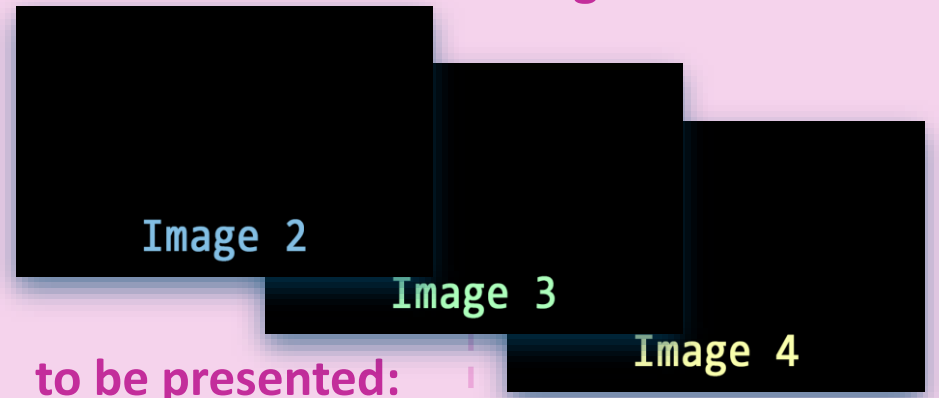
FIFO

Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

Swap Chain

available images:



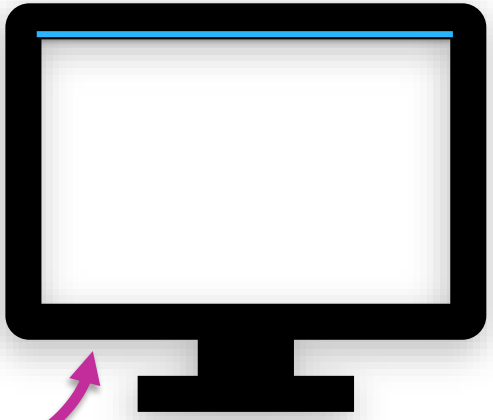
to be presented:

presented image:

"FIFO"

Presentation Mode

"vertical blank"



Application is lagging behind

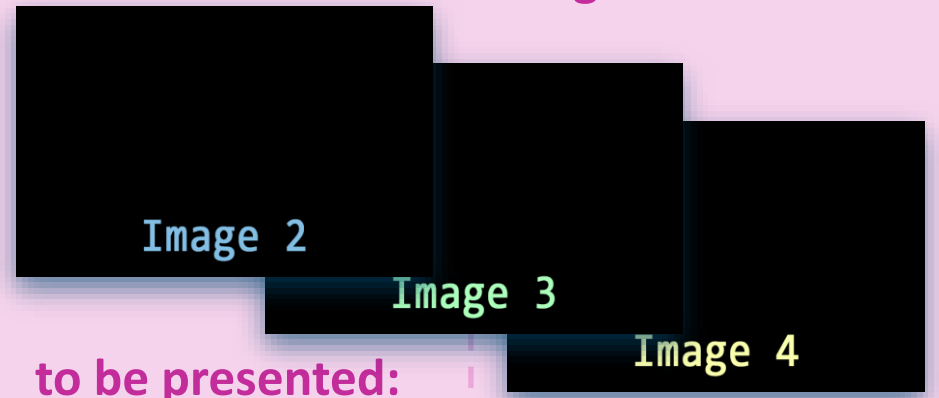
FIFO Relaxed

Application/Render Loop

```
while (true) {

  acquireNextImage();


  draw();


  present();

}
```

Swap Chain

available images:



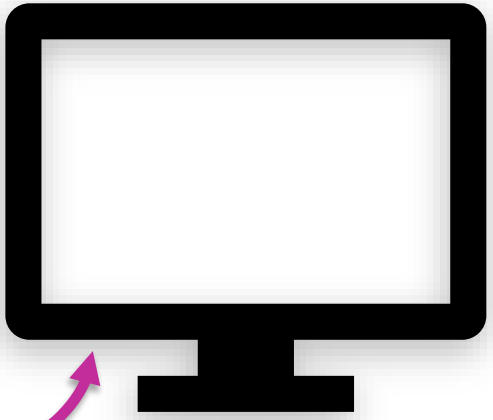
to be presented:

presented image:

"FIFO Relaxed"

Presentation Mode

"vertical blank"



Application is lagging behind

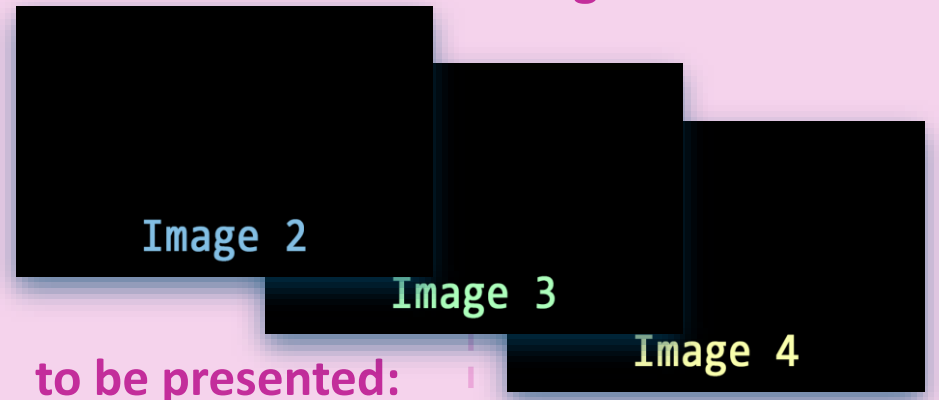
FIFO Relaxed

Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();

  present();

}
```

Swap Chain

available images:



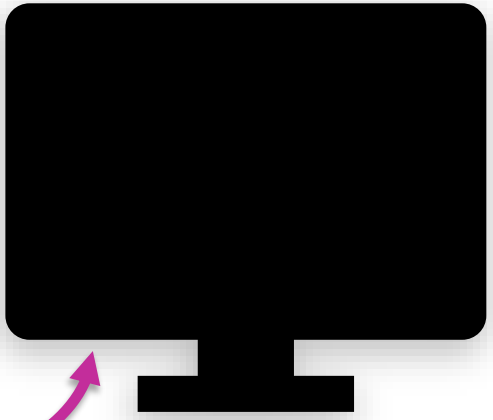
to be presented:

presented image:

" FIFO Relaxed "

Presentation Mode

"vertical blank"



Application is lagging behind

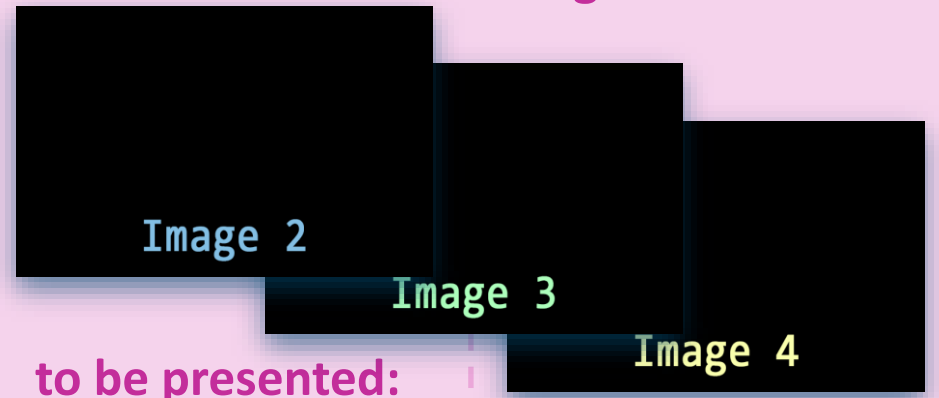
FIFO Relaxed

Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

Swap Chain

available images:



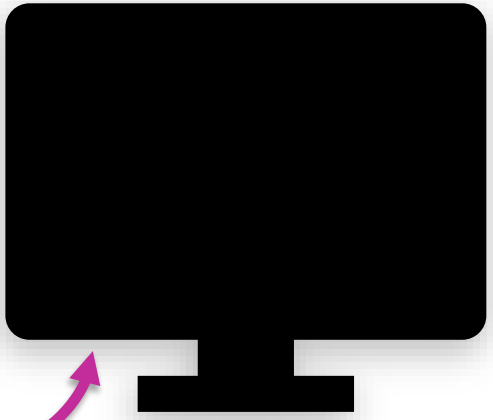
to be presented:

presented image:

" FIFO Relaxed "

Presentation Mode

"vertical blank"



Application is lagging behind

FIFO Relaxed

Application/Render Loop

Swap Chain

" FIFO Relaxed "

Presentation Mode

VK_PRESENT_MODE_FIFO_RELAXED_KHR specifies that the presentation engine generally waits for the next vertical blanking period to update the current image. If a vertical blanking period has already passed since the last update of the current image then the presentation engine does not wait for another vertical blanking period for the update, meaning this mode **may** result in visible tearing in this case. This mode is useful for reducing visual stutter with an application that will mostly present a new image before the next vertical blanking period, but may occasionally be late, and present a new image just after the next vertical blanking period.

The Khronos Group. Vulkan 1.2.196 Specificaton

```
whi
  a
  d
  present();
}
```

Application is lagging behind

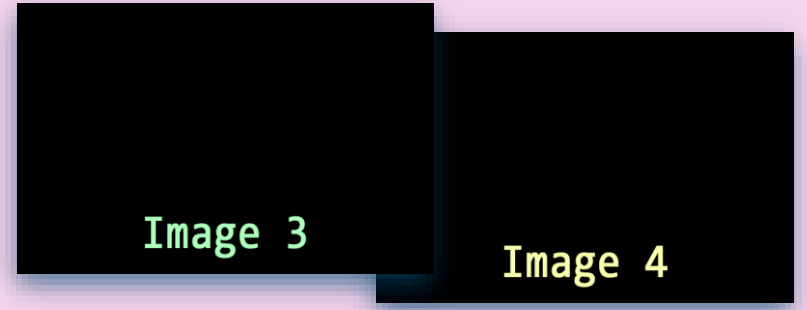
Mailbox

Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

Swap Chain

available images:



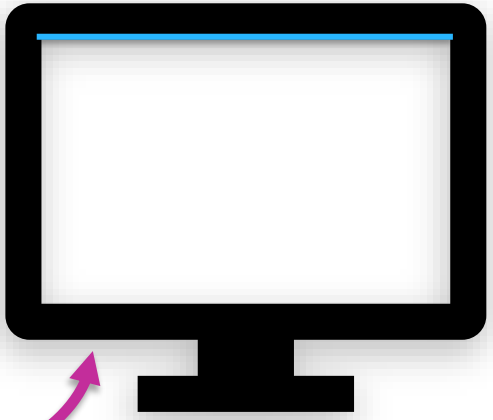
to be presented:

presented image:

"Mailbox"

Presentation Mode

"vertical blank"



Mailbox

Application/Render Loop

```
while (true) {

  acquireNextImage();

  draw();

  present();

}
```

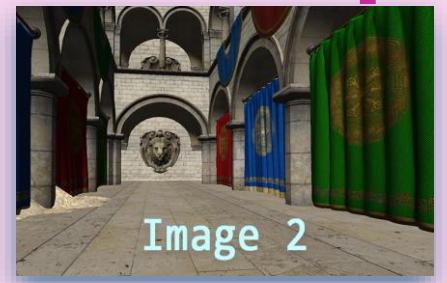
Image 3

Swap Chain

available images:

Image 4

to be presented:

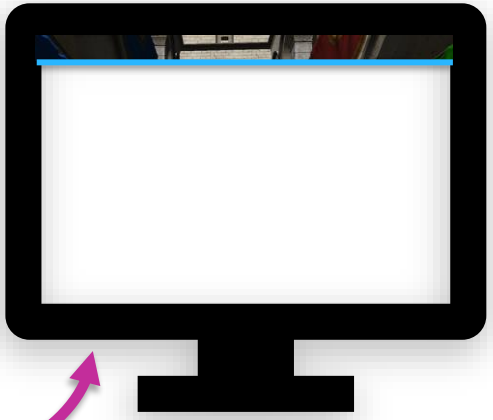


presented image:

"Mailbox"

Presentation Mode

"vertical blank"



Mailbox

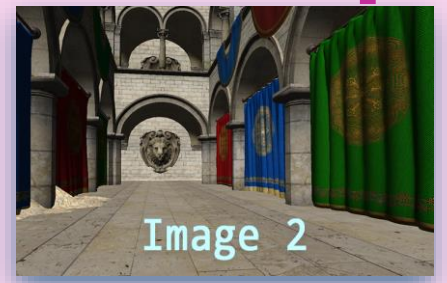
Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

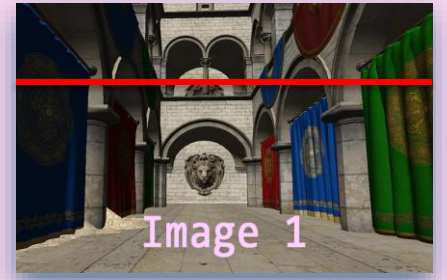
Swap Chain

available images:

to be presented:



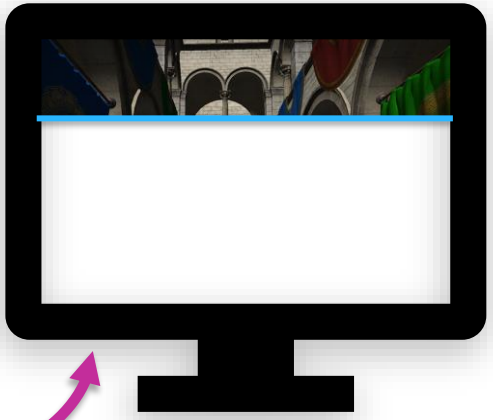
presented image:



"Mailbox"

Presentation Mode

"vertical blank"



Application is running ahead

Mailbox

Application/Render Loop

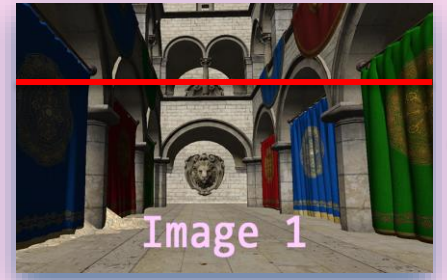
```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

Swap Chain

available images:

to be presented:

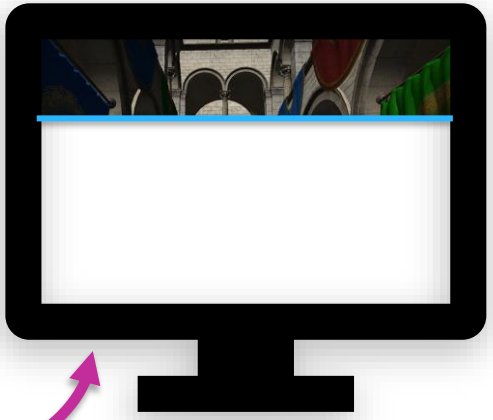
presented image:



"Mailbox"

Presentation Mode

"vertical blank"



Application is running ahead

Mailbox

Application/Render Loop

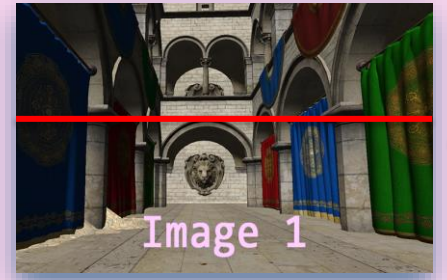
```
while (true) {

  acquireNextImage();

  draw();

  present();

}
```

Swap Chain

available images:

to be presented:

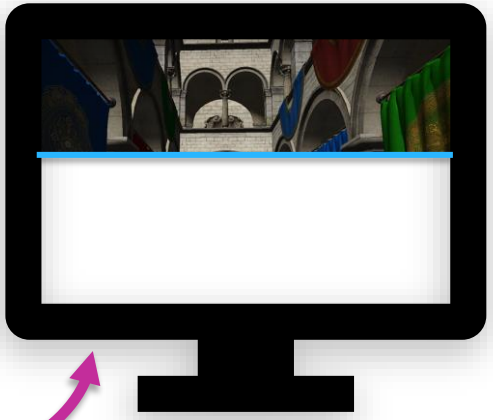
presented image:



"Mailbox"

Presentation Mode

"vertical blank"



Application is running ahead

Mailbox

Application/Render Loop

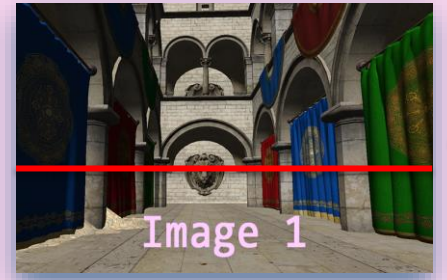
```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

Swap Chain

available images:

to be presented:

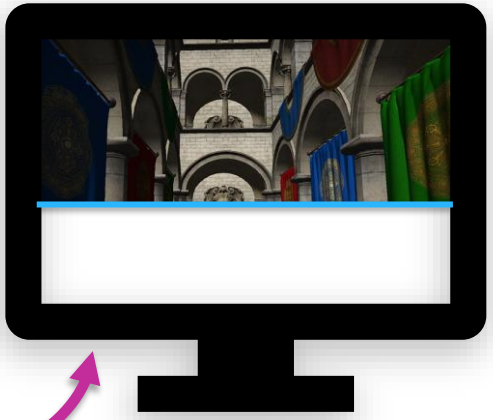
presented image:



"Mailbox"

Presentation Mode

"vertical blank"



Application is running ahead

Mailbox

Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

Swap Chain

available images:

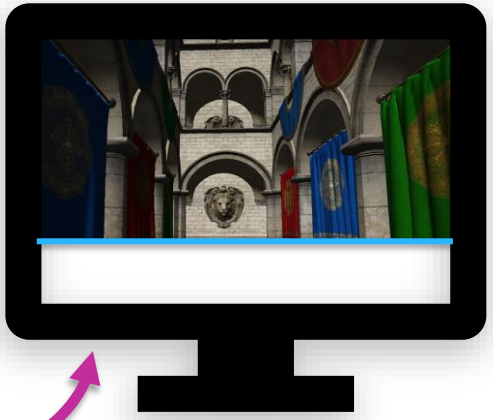
to be presented:

presented image:

"Mailbox"

Presentation Mode

"vertical blank"



Application is running ahead

Mailbox

Application/Render Loop

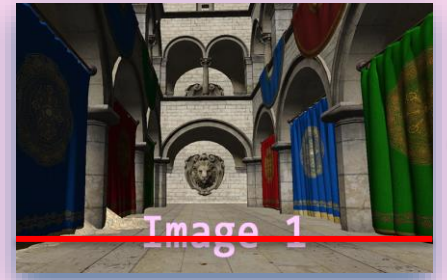
```
while (true) {

  acquireNextImage();


  draw();


  present();

}
```

Swap Chain

available images:

to be presented:

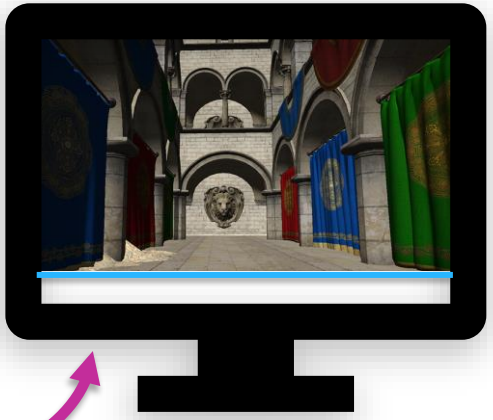
presented image:



"Mailbox"

Presentation Mode

"vertical blank"



Application is running ahead

Mailbox

Application/Render Loop

```
while (true) {

  acquireNextImage();


  draw();


  present();

}
```

Swap Chain

available images:

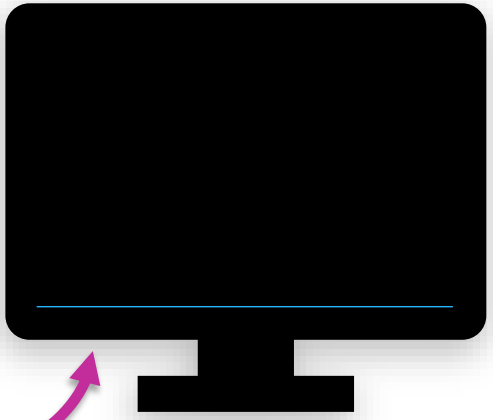
to be presented:

presented image:

"Mailbox"

Presentation Mode

"vertical blank"



Application is running ahead

Mailbox

Application/Render Loop

```
while (

  acqui

  draw(

  present();

}
```

Swap Chain

available images:

"Mailbox"

Presentation Mode

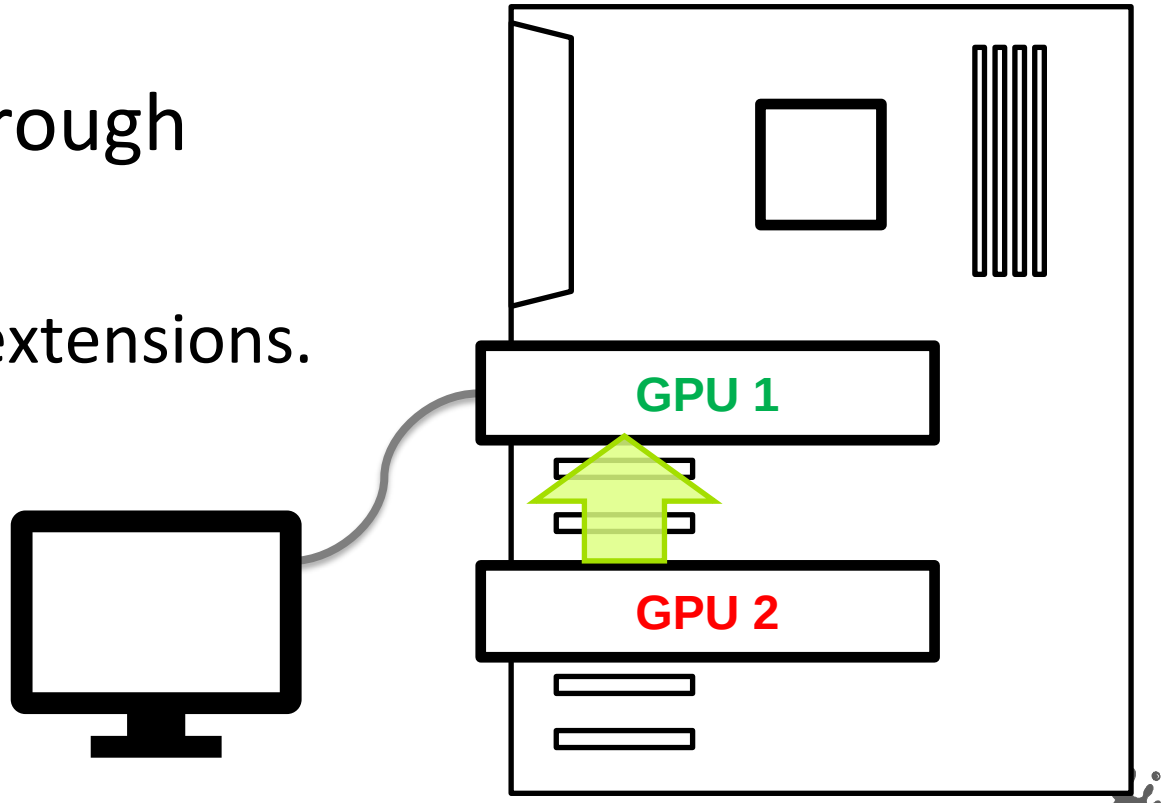
VK_PRESENT_MODE_MAILBOX_KHR specifies that the presentation engine waits for the next vertical blanking period to update the current image. Tearing **cannot** be observed. An internal single-entry queue is used to hold pending presentation requests. If the queue is full when a new presentation request is received, the new request replaces the existing entry, and any images associated with the prior entry become available for re-use by the application. One request is removed from the queue and processed during each vertical blanking period in which the queue is non-empty.

The Khronos Group. Vulkan 1.2.196 Specificaton

Application is running ahead

Extensions

- Swapchain extension: `"VK_KHR_swapchain"`
  - Device-level extension
  - Companion to the instance-level extension `"VK_KHR_surface"`
    i.e., it requires `"VK_KHR_surface"`
- No need to handle presentation through the correct device manually
  - Handled transparently through the extensions.
  - Just enable extensions,
  - and call `vkQueuePresentKHR`,
  - -> presentation engine.



Extensions

- Device must provide queue families with presentation support
  - … to a particular surface!
- Can be queried after `"VK_KHR_surface"` has been enabled.
  - Use `vkGetPhysicalDeviceSurfaceSupportKHR`
    - Pass queue family index
    - Pass `VkSurfaceKHR` handle

Code

Application/Render Loop

```
while (true) {

  acquireNextImage();




  draw();




  present();

}
```

Code

Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

```
VkSurfaceKHR surface = ...
// ^ use GLFW, or search for examples about "window
//             system integration" (short: "WSI")

// ...

VkSwapchainCreateInfoKHR createInfo = {};
createInfo.sType = VK_STRUCTURE_TYPE_SWAPCHAIN_CREATE_INFO_KHR;
createInfo.surface = surface;
createInfo.minImageCount = 4;
createInfo.imageFormat = VK_FORMAT_R8G8B8A8_SRGB;
createInfo.imageColorSpace = VK_COLOR_SPACE_SRGB_NONLINEAR_KHR;
// Use vkGetPhysicalDeviceSurfaceFormatsKHR to find out ^
createInfo.imageExtent = VkExtent2D{ 1920, 1080 };
createInfo.imageArrayLayers = 1;
createInfo.imageUsage = VK_IMAGE_USAGE_COLOR_ATTACHMENT_BIT;
createInfo.presentMode = VK_PRESENT_MODE_IMMEDIATE_KHR;
// further settings: images queue ownership, alpha, ...

vkCreateSwapchainKHR(..., &createInfo, ...);
```

Code

Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

```
VkSurfaceKHR surface = ...
// ^ use GLFW, or search for examples about "window
//             system integration" (short: "WSI")

// ...

VkSwapchainCreateInfoKHR createInfo = {};
createInfo.sType = VK_STRUCTURE_TYPE_SWAPCHAIN_CREATE_INFO_KHR;
createInfo.surface = surface;
createInfo.minImageCount = 4;
createInfo.imageFormat = VK_FORMAT_R8G8B8A8_SRGB;
createInfo.imageColorSpace = VK_COLOR_SPACE_SRGB_NONLINEAR_KHR;
// Use vkGetPhysicalDeviceSurfaceFormatsKHR to find out ^
createInfo.imageExtent = VkExtent2D{ 1920, 1080 };
createInfo.imageArrayLayers = 1;
createInfo.imageUsage = VK_IMAGE_USAGE_COLOR_ATTACHMENT_BIT;
createInfo.presentMode = VK_PRESENT_MODE_IMMEDIATE_KHR;
// further settings: images queue ownership, alpha, ...

vkCreateSwapchainKHR(..., &createInfo, ...);
```

Code

Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

```
VkSurfaceKHR surface = ...
// ^ use GLFW, or search for examples about "window
//             system integration" (short: "WSI")

// ...

VkSwapchainCreateInfoKHR createInfo = {};
createInfo.sType = VK_STRUCTURE_TYPE_SWAPCHAIN_CREATE_INFO_KHR;
createInfo.surface = surface;
createInfo.minImageCount = 4;
createInfo.imageFormat = VK_FORMAT_R8G8B8A8_SRGB;
createInfo.imageColorSpace = VK_COLOR_SPACE_SRGB_NONLINEAR_KHR;
// Use vkGetPhysicalDeviceSurfaceFormatsKHR to find out ^
createInfo.imageExtent = VkExtent2D{ 1920, 1080 };
createInfo.imageArrayLayers = 1;
createInfo.imageUsage = VK_IMAGE_USAGE_COLOR_ATTACHMENT_BIT;
createInfo.presentMode = VK_PRESENT_MODE_IMMEDIATE_KHR;
// further settings: images queue ownership, alpha, ...

vkCreateSwapchainKHR(..., &createInfo, ...);
```

Code

Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

```
VkSurfaceKHR surface = ...
// ^ use GLFW, or search for examples about "window
//             system integration" (short: "WSI")

// ...

VkSwapchainCreateInfoKHR createInfo = {};
createInfo.sType = VK_STRUCTURE_TYPE_SWAPCHAIN_CREATE_INFO_KHR;
createInfo.surface = surface;
createInfo.minImageCount = 4;
createInfo.imageFormat = VK_FORMAT_R8G8B8A8_SRGB;
createInfo.imageColorSpace = VK_COLOR_SPACE_SRGB_NONLINEAR_KHR;
// Use vkGetPhysicalDeviceSurfaceFormatsKHR to find out ^
createInfo.imageExtent = VkExtent2D{ 1920, 1080 };
createInfo.imageArrayLayers = 1;
createInfo.imageUsage = VK_IMAGE_USAGE_COLOR_ATTACHMENT_BIT;
createInfo.presentMode = VK_PRESENT_MODE_IMMEDIATE_KHR;
// further settings: images queue ownership, alpha, ...

vkCreateSwapchainKHR(..., &createInfo, ...);
```

Code

Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

```
VkSurfaceKHR surface = ...
// ^ use GLFW, or search for examples about "window
//             system integration" (short: "WSI")

// ...

VkSwapchainCreateInfoKHR createInfo = {};
createInfo.sType = VK_STRUCTURE_TYPE_SWAPCHAIN_CREATE_INFO_KHR;
createInfo.surface = surface;
createInfo.minImageCount = 4;
createInfo.imageFormat = VK_FORMAT_R8G8B8A8_SRGB;
createInfo.imageColorSpace = VK_COLOR_SPACE_SRGB_NONLINEAR_KHR;
// Use vkGetPhysicalDeviceSurfaceFormatsKHR to find out ^
createInfo.imageExtent = VkExtent2D{ 1920, 1080 };
createInfo.imageArrayLayers = 1;
createInfo.imageUsage = VK_IMAGE_USAGE_COLOR_ATTACHMENT_BIT;
createInfo.presentMode = VK_PRESENT_MODE_IMMEDIATE_KHR;
// further settings: images queue ownership, alpha, ...

vkCreateSwapchainKHR(..., &createInfo, ...);
```

Code

Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

```
VkSurfaceKHR surface = ...
// ^ use GLFW, or search for examples about "window
//             system integration" (short: "WSI")

// ...

VkSwapchainCreateInfoKHR createInfo = {};
createInfo.sType = VK_STRUCTURE_TYPE_SWAPCHAIN_CREATE_INFO_KHR;
createInfo.surface = surface;
createInfo.minImageCount = 4;
createInfo.imageFormat = VK_FORMAT_R8G8B8A8_SRGB;
createInfo.imageColorSpace = VK_COLOR_SPACE_SRGB_NONLINEAR_KHR;
// Use vkGetPhysicalDeviceSurfaceFormatsKHR to find out ^
createInfo.imageExtent = VkExtent2D{ 1920, 1080 };
createInfo.imageArrayLayers = 1;
createInfo.imageUsage = VK_IMAGE_USAGE_COLOR_ATTACHMENT_BIT;
createInfo.presentMode = VK_PRESENT_MODE_IMMEDIATE_KHR;
// further settings: images queue ownership, alpha, ...

vkCreateSwapchainKHR(..., &createInfo, ...);
```

Code

Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

```
VkSurfaceKHR surface = ...
// ^ use GLFW, or search for examples about "window
//             system integration" (short: "WSI")

// ...

VkSwapchainCreateInfoKHR createInfo = {};
createInfo.sType = VK_STRUCTURE_TYPE_SWAPCHAIN_CREATE_INFO_KHR;
createInfo.surface = surface;
createInfo.minImageCount = 4;
createInfo.imageFormat = VK_FORMAT_R8G8B8A8_SRGB;
createInfo.imageColorSpace = VK_COLOR_SPACE_SRGB_NONLINEAR_KHR;
// Use vkGetPhysicalDeviceSurfaceFormatsKHR to find out ^
createInfo.imageExtent = VkExtent2D{ 1920, 1080 };
createInfo.imageArrayLayers = 1;
createInfo.imageUsage = VK_IMAGE_USAGE_COLOR_ATTACHMENT_BIT;
createInfo.presentMode = VK_PRESENT_MODE_IMMEDIATE_KHR;
// further settings: images queue ownership, alpha, ...

vkCreateSwapchainKHR(..., &createInfo, ...);
```

Code

Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

```
VkSurfaceKHR surface = ...
// ^ use GLFW, or search for examples about "window
//             system integration" (short: "WSI")

// ...

VkSwapchainCreateInfoKHR createInfo = {};
createInfo.sType = VK_STRUCTURE_TYPE_SWAPCHAIN_CREATE_INFO_KHR;
createInfo.surface = surface;
createInfo.minImageCount = 4;
createInfo.imageFormat = VK_FORMAT_R8G8B8A8_SRGB;
createInfo.imageColorSpace = VK_COLOR_SPACE_SRGB_NONLINEAR_KHR;
// Use vkGetPhysicalDeviceSurfaceFormatsKHR to find out ^
createInfo.imageExtent = VkExtent2D{ 1920, 1080 };
createInfo.imageArrayLayers = 1;
createInfo.imageUsage = VK_IMAGE_USAGE_COLOR_ATTACHMENT_BIT;
createInfo.presentMode = VK_PRESENT_MODE_IMMEDIATE_KHR;
// further settings: images queue ownership, alpha, ...

vkCreateSwapchainKHR(..., &createInfo, ...);
```

Code

Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

```
VkSurfaceKHR surface = ...
// ^ use GLFW, or search for examples about "window
//             system integration" (short: "WSI")

// ...

VkSwapchainCreateInfoKHR createInfo = {};
createInfo.sType = VK_STRUCTURE_TYPE_SWAPCHAIN_CREATE_INFO_KHR;
createInfo.surface = surface;
createInfo.minImageCount = 4;
createInfo.imageFormat = VK_FORMAT_R8G8B8A8_SRGB;
createInfo.imageColorSpace = VK_COLOR_SPACE_SRGB_NONLINEAR_KHR;
// Use vkGetPhysicalDeviceSurfaceFormatsKHR to find out ^
createInfo.imageExtent = VkExtent2D{ 1920, 1080 };
createInfo.imageArrayLayers = 1;
createInfo.imageUsage = VK_IMAGE_USAGE_COLOR_ATTACHMENT_BIT;
createInfo.presentMode = VK_PRESENT_MODE_IMMEDIATE_KHR;
// further settings: images queue ownership, alpha, ...

vkCreateSwapchainKHR(..., &createInfo, ...);
```

Code

Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

```
VkSurfaceKHR surface = ...
// ^ use GLFW, or search for examples about "window
//             system integration" (short: "WSI")

// ...

VkSwapchainCreateInfoKHR createInfo = {};
createInfo.sType = VK_STRUCTURE_TYPE_SWAPCHAIN_CREATE_INFO_KHR;
createInfo.surface = surface;
createInfo.minImageCount = 4;
createInfo.imageFormat = VK_FORMAT_R8G8B8A8_SRGB;
createInfo.imageColorSpace = VK_COLOR_SPACE_SRGB_NONLINEAR_KHR;
// Use vkGetPhysicalDeviceSurfaceFormatsKHR to find out ^
createInfo.imageExtent = VkExtent2D{ 1920, 1080 };
createInfo.imageArrayLayers = 1;
createInfo.imageUsage = VK_IMAGE_USAGE_COLOR_ATTACHMENT_BIT;
createInfo.presentMode = VK_PRESENT_MODE_IMMEDIATE_KHR;
// further settings: images queue ownership, alpha, ...

vkCreateSwapchainKHR(..., &createInfo, ...);
```

Code

Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

```
VkDevice device;
VkSwapchainKHR swapchain;

// Query how many swapchain images we got:
uint32_t count;
vkGetSwapchainImagesKHR(device, swapchain, &count, nullptr);

// Retrieve the swapchain image handles:
VkImage* swapchainImageHandles = new VkImage[count];
vkGetSwapchainImagesKHR(device, swapchain,
                        &count, swapchainImageHandles);
```

Code

Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

```
VkDevice device;
VkSwapchainKHR swapchain;

// Query how many swapchain images we got:
uint32_t count;
vkGetSwapchainImagesKHR(device, swapchain, &count, nullptr);

// Retrieve the swapchain image handles:
VkImage* swapchainImageHandles = new VkImage[count];
vkGetSwapchainImagesKHR(device, swapchain,
                        &count, swapchainImageHandles);
```

Code

Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

```
VkDevice device;
VkSwapchainKHR swapchain;

// Query how many swapchain images we got:
uint32_t count;
vkGetSwapchainImagesKHR(device, swapchain, &count, nullptr);

// Retrieve the swapchain image handles:
VkImage* swapchainImageHandles = new VkImage[count];
vkGetSwapchainImagesKHR(device, swapchain,
                        &count, swapchainImageHandles);
```

Code

Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

```
VkDevice device;
VkSwapchainKHR swapchain;
VkSemaphore imageAvailableSemaphore;
VkFence imageAvailableFence;
uint32_t imageIndex;

vkAcquireNextImageKHR(
  device, swapchain,
  UINT64_MAX,                // <-- timeout
  imageAvailableSemaphore, // <-- signal when acquired
  imageAvailableFence,     // <-- signal when acquired
  &imageIndex                // <-- output parameter
);
```

Code

Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

```
VkDevice device;
VkSwapchainKHR swapchain;
VkSemaphore imageAvailableSemaphore;
VkFence imageAvailableFence;
uint32_t imageIndex;

vkAcquireNextImageKHR(
  device, swapchain,
  UINT64_MAX,                // <-- timeout
  imageAvailableSemaphore, // <-- signal when acquired
  imageAvailableFence,     // <-- signal when acquired
  &imageIndex              // <-- output parameter
);
```

Code

Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

```
VkDevice device;
VkSwapchainKHR swapchain;
VkSemaphore imageAvailableSemaphore;
VkFence imageAvailableFence;
uint32_t imageIndex;

vkAcquireNextImageKHR(
  device, swapchain,
  UINT64_MAX,                // <-- timeout
  imageAvailableSemaphore,   // <-- signal when acquired
  imageAvailableFence,       // <-- signal when acquired
  &imageIndex                // <-- output parameter
);
```

Code

Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

```
VkDevice device;
VkSwapchainKHR swapchain;
VkSemaphore imageAvailableSemaphore;
VkFence imageAvailableFence;
uint32_t imageIndex;

vkAcquireNextImageKHR(
  device, swapchain,
  UINT64_MAX,                // <-- timeout
  imageAvailableSemaphore,   // <-- signal when acquired
  imageAvailableFence,       // <-- signal when acquired
  &imageIndex                // <-- output parameter
);
```

Code

Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

```
VkDevice device;
VkSwapchainKHR swapchain;
VkSemaphore imageAvailableSemaphore;
VkFence imageAvailableFence;
uint32_t imageIndex;

vkAcquireNextImageKHR(
  device, swapchain,
  UINT64_MAX,                // <-- timeout
  imageAvailableSemaphore, // <-- signal when acquired
  imageAvailableFence,     // <-- signal when acquired
  &imageIndex                // <-- output parameter
);
```

```
VkImage* swapchainImageHandles = new VkImage[count];
vkGetSwapchainImagesKHR(device, swapchain,
                        &count,
swapchainImageHandles);
```

Code

Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

```
VkQueue queue;
VkSemaphore renderFinishedSemaphore;
VkFence syncHostWithDeviceFence;

VkSubmitInfo submitInfo = {};
submitInfo.sType = VK_STRUCTURE_TYPE_SUBMIT_INFO;
submitInfo.commandBufferCount = ...
submitInfo.pCommandBuffers = ...
submitInfo.waitSemaphoreCount = 1;
submitInfo.pWaitSemaphores = &imageAvailableSemaphore;
submitInfo.signalSemaphoreCount = 1;
submitInfo.pSignalSemaphores = &renderFinishedSemaphore;

vkQueueSubmit(queue, 1, &submitInfo, syncHostWithDeviceFence);
```

Code

Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

```
VkQueue queue;
VkSemaphore renderFinishedSemaphore;
VkFence syncHostWithDeviceFence;

VkSubmitInfo submitInfo = {};
submitInfo.sType = VK_STRUCTURE_TYPE_SUBMIT_INFO;
submitInfo.commandBufferCount = ...
submitInfo.pCommandBuffers = ...
submitInfo.waitSemaphoreCount = 1;
submitInfo.pWaitSemaphores = &imageAvailableSemaphore;
submitInfo.signalSemaphoreCount = 1;
submitInfo.pSignalSemaphores = &renderFinishedSemaphore;

vkQueueSubmit(queue, 1, &submitInfo, syncHostWithDeviceFence);
```

Code

Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

```
VkQueue queue;
VkSemaphore renderFinishedSemaphore;
VkFence syncHostWithDeviceFence;

VkSubmitInfo submitInfo = {};
submitInfo.sType = VK_STRUCTURE_TYPE_SUBMIT_INFO;
submitInfo.commandBufferCount = ...
submitInfo.pCommandBuffers = ...
submitInfo.waitSemaphoreCount = 1;
submitInfo.pWaitSemaphores = &imageAvailableSemaphore;
submitInfo.signalSemaphoreCount = 1;
submitInfo.pSignalSemaphores = &renderFinishedSemaphore;

vkQueueSubmit(queue, 1, &submitInfo, syncHostWithDeviceFence);
```

Code

Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

```
VkQueue queue;
VkSemaphore renderFinishedSemaphore;
VkFence syncHostWithDeviceFence;

VkSubmitInfo submitInfo = {};
submitInfo.sType = VK_STRUCTURE_TYPE_SUBMIT_INFO;
submitInfo.commandBufferCount = ...
submitInfo.pCommandBuffers = ...
submitInfo.waitSemaphoreCount = 1;
submitInfo.pWaitSemaphores = &imageAvailableSemaphore;
submitInfo.signalSemaphoreCount = 1;
submitInfo.pSignalSemaphores = &renderFinishedSemaphore;

vkQueueSubmit(queue, 1, &submitInfo, syncHostWithDeviceFence);
```

Code

Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

```
VkQueue queue;
VkSemaphore renderFinishedSemaphore;
VkFence syncHostWithDeviceFence;

VkSubmitInfo submitInfo = {};
submitInfo.sType = VK_STRUCTURE_TYPE_SUBMIT_INFO;
submitInfo.commandBufferCount = ...
submitInfo.pCommandBuffers = ...
submitInfo.waitSemaphoreCount = 1;
submitInfo.pWaitSemaphores = &imageAvailableSemaphore;
submitInfo.signalSemaphoreCount = 1;
submitInfo.pSignalSemaphores = &renderFinishedSemaphore;

vkQueueSubmit(queue, 1, &submitInfo, syncHostWithDeviceFence);
```

Code

Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

```
VkQueue queue;
VkSemaphore renderFinishedSemaphore;
VkFence syncHostWithDeviceFence;

VkSubmitInfo submitInfo = {};
submitInfo.sType = VK_STRUCTURE_TYPE_SUBMIT_INFO;
submitInfo.commandBufferCount = ...
submitInfo.pCommandBuffers = ...
submitInfo.waitSemaphoreCount = 1;
submitInfo.pWaitSemaphores = &imageAvailableSemaphore;
submitInfo.signalSemaphoreCount = 1;
submitInfo.pSignalSemaphores = &renderFinishedSemaphore;

vkQueueSubmit(queue, 1, &submitInfo, syncHostWithDeviceFence);
```

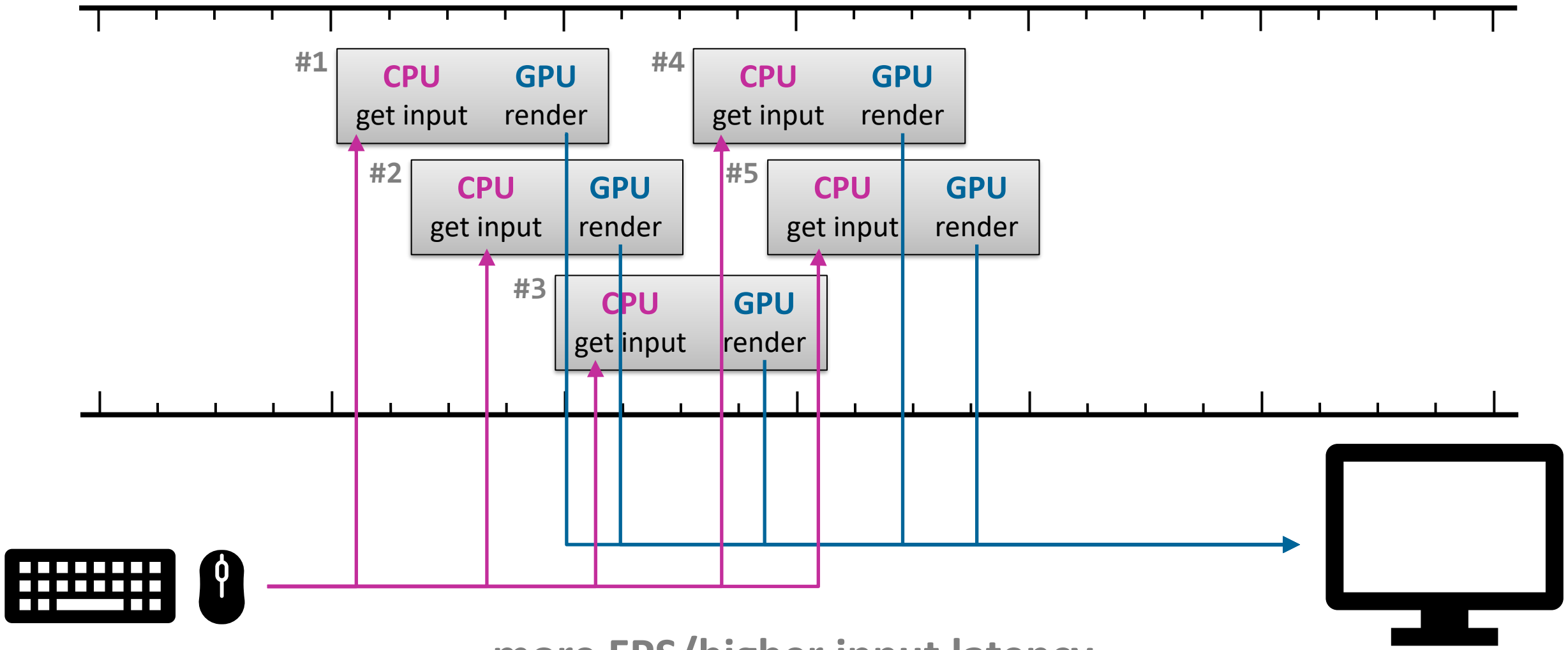
Code

Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

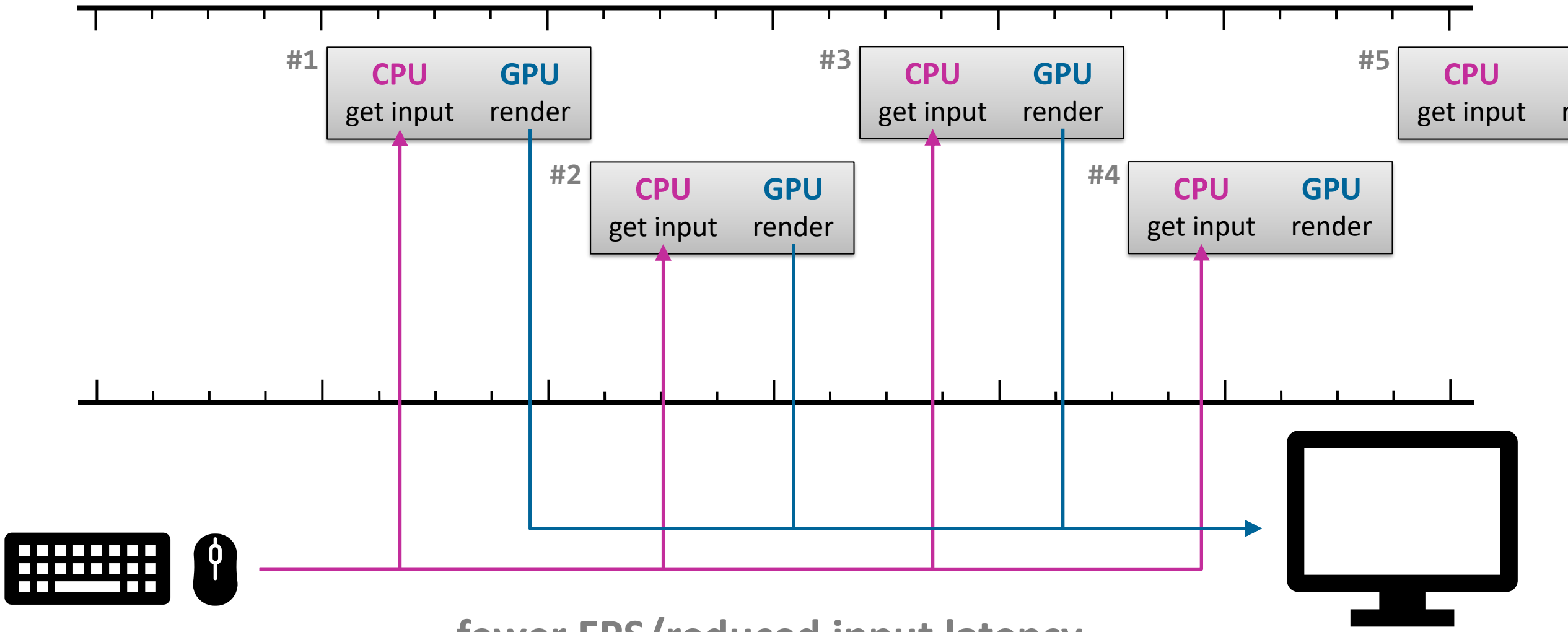
```
VkQueue queue;
VkSemaphore renderFinishedSemaphore;
VkFence syncHostWithDeviceFence;

VkSubmitInfo submitInfo = {};
submitInfo.sType = VK_STRUCTURE_TYPE_SUBMIT_INFO;
submitInfo.commandBufferCount = ...
submitInfo.pCommandBuffers = ...
submitInfo.waitSemaphoreCount = 1;
submitInfo.pWaitSemaphores = &imageAvailableSemaphore;
submitInfo.signalSemaphoreCount = 1;
submitInfo.pSignalSemaphores = &renderFinishedSemaphore;

vkQueueSubmit(queue, 1, &submitInfo, syncHostWithDeviceFence);
```

High FPS vs. Input Lag



more FPS/higher input latency

Input Lag Reduction



fewer FPS/reduced input latency

Input Lag Reduction

- 2019: AMD's Radeon™ Anti-Lag; NVIDIA's Ultra-Low Latency Mode
  - Reduces number of pre-rendered frames/frames in flight
  - Trades FPS for reduced input latency
- OpenGL: Handled by the **driver**
- Vulkan: Handled by the **programmer**

Code

Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

```
VkQueue queue;
VkSemaphore renderFinishedSemaphore;
VkFence syncHostWithDeviceFence;

VkSubmitInfo submitInfo = {};
submitInfo.sType = VK_STRUCTURE_TYPE_SUBMIT_INFO;
submitInfo.commandBufferCount = ...
submitInfo.pCommandBuffers = ...
submitInfo.waitSemaphoreCount = 1;
submitInfo.pWaitSemaphores = &imageAvailableSemaphore;
submitInfo.signalSemaphoreCount = 1;
submitInfo.pSignalSemaphores = &renderFinishedSemaphore;

vkQueueSubmit(queue, 1, &submitInfo, syncHostWithDeviceFence);
```

Code

Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

```
VkQueue queue;
VkSemaphore renderFinishedSemaphore;
VkFence syncHostWithDeviceFence;

VkSubmitInfo submitInfo = {};
submitInfo.sType = VK_STRUCTURE_TYPE_SUBMIT_INFO;
submitInfo.commandBufferCount = ...
submitInfo.pCommandBuffers = ...
submitInfo.waitSemaphoreCount = 1;
submitInfo.pWaitSemaphores = &imageAvailableSemaphore;
submitInfo.signalSemaphoreCount = 1;
submitInfo.pSignalSemaphores = &renderFinishedSemaphore;

vkQueueSubmit(queue, 1, &submitInfo, syncHostWithDeviceFence);
```

Code

Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

```
VkPresentInfoKHR presentInfo = {};
presentInfo.sType = VK_STRUCTURE_TYPE_PRESENT_INFO_KHR;
presentInfo.waitSemaphoreCount = 1;
presentInfo.pWaitSemaphores = &renderFinishedSemaphore;
presentInfo.swapchainCount = 1;
presentInfo.pSwapchains = &swapchain;
presentInfo.pImageIndices = &imageIndex;

vkQueuePresentKHR(queue, &presentInfo);
```

Code

Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

```
VkPresentInfoKHR presentInfo = {};
presentInfo.sType = VK_STRUCTURE_TYPE_PRESENT_INFO_KHR;
presentInfo.waitSemaphoreCount = 1;
presentInfo.pWaitSemaphores = &renderFinishedSemaphore;
presentInfo.swapchainCount = 1;
presentInfo.pSwapchains = &swapchain;
presentInfo.pImageIndices = &imageIndex;

vkQueuePresentKHR(queue, &presentInfo);
```

Code

Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

```
VkPresentInfoKHR presentInfo = {};
presentInfo.sType = VK_STRUCTURE_TYPE_PRESENT_INFO_KHR;
presentInfo.waitSemaphoreCount = 1;
presentInfo.pWaitSemaphores = &renderFinishedSemaphore;
presentInfo.swapchainCount = 1;
presentInfo.pSwapchains = &swapchain;
presentInfo.pImageIndices = &imageIndex;

vkQueuePresentKHR(queue, &presentInfo);
```

Code

Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

```
VkPresentInfoKHR presentInfo = {};
presentInfo.sType = VK_STRUCTURE_TYPE_PRESENT_INFO_KHR;
presentInfo.waitSemaphoreCount = 1;
presentInfo.pWaitSemaphores = &renderFinishedSemaphore;
presentInfo.swapchainCount = 1;
presentInfo.pSwapchains = &swapchain;
presentInfo.pImageIndices = &imageIndex;

vkQueuePresentKHR(queue, &presentInfo);
```

The Swap Chain With a Fast CPU

Application/Render Loop

```
while (true) {

  acquireNextImage();

  draw();

  present();

}
```

Image 1

Swap Chain

available images:

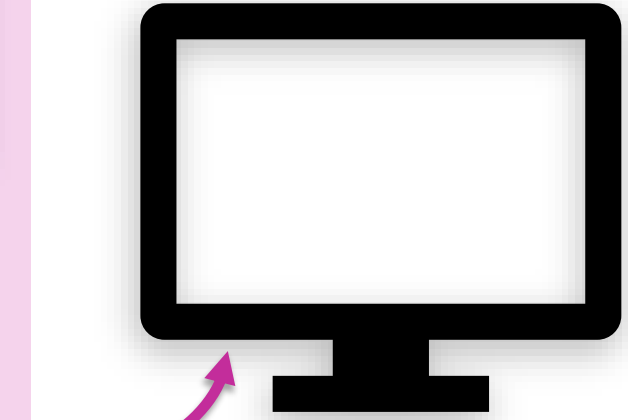
Image 2

Image 3

Image 4

to be presented:

presented image:



The Swap Chain With a Fast CPU

Application/Render Loop

```
while (true) {

  acquireNextImage();


  draw();


  present();

}
```

Image 1

Swap Chain

available images:

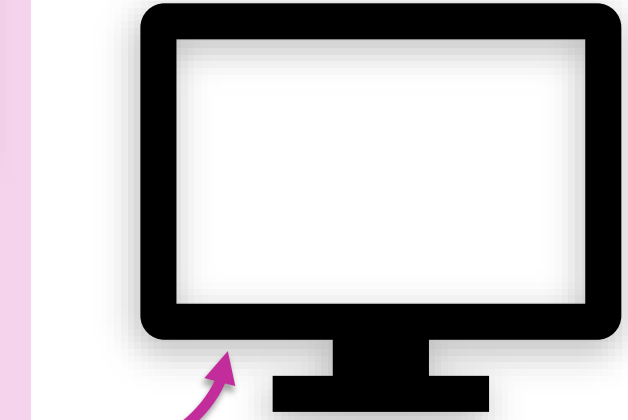
Image 2

Image 3

Image 4

to be presented:

presented image:



The Swap Chain With a Fast CPU

Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

Image 1

Swap Chain

available images:

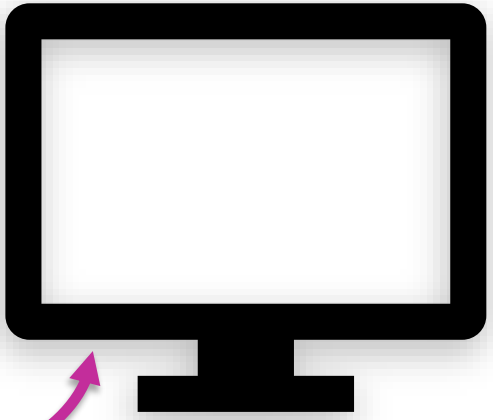
Image 2

Image 3

Image 4

to be presented:

presented image:



The Swap Chain With a Fast CPU

Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

Swap Chain

available images:

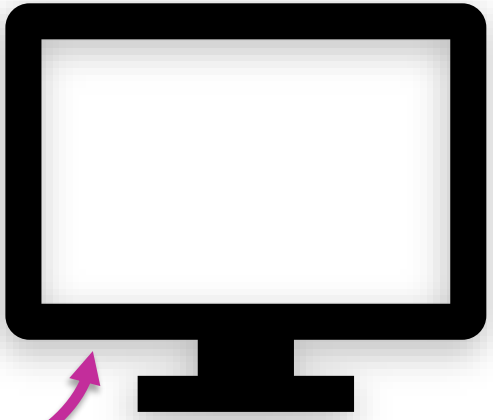
Image 2

Image 3

Image 4

to be presented:

presented image:



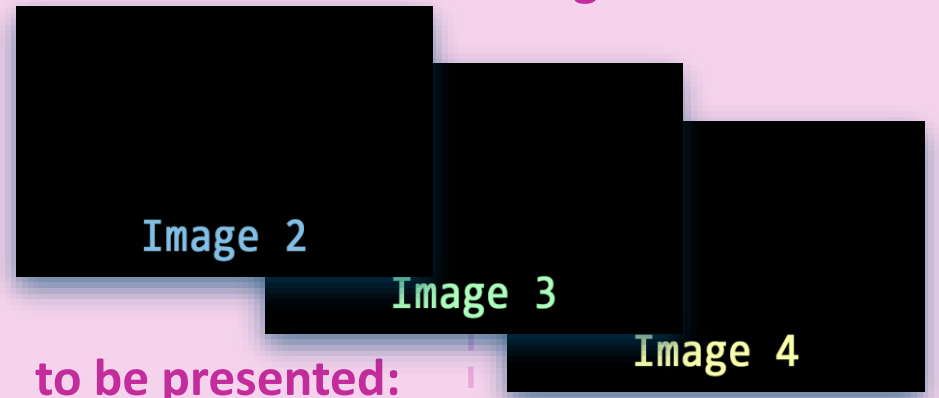
The Swap Chain With a Fast CPU

Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

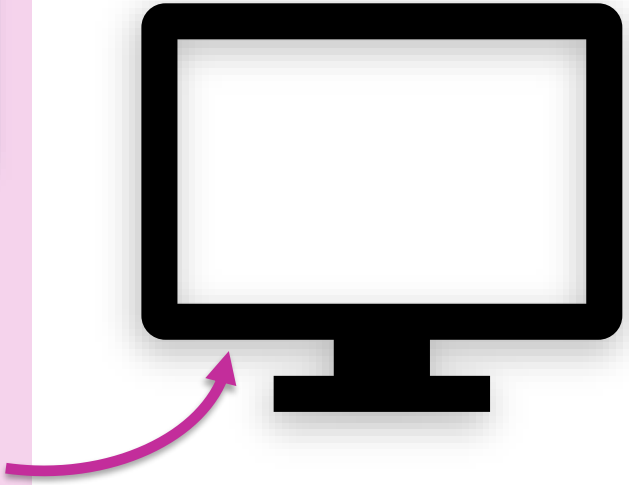
Swap Chain

available images:



to be presented:

presented image:



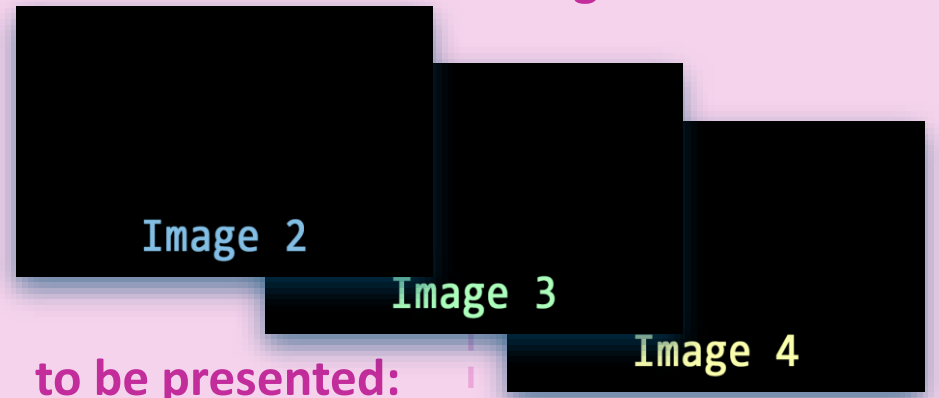
The Swap Chain With a Fast CPU

Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

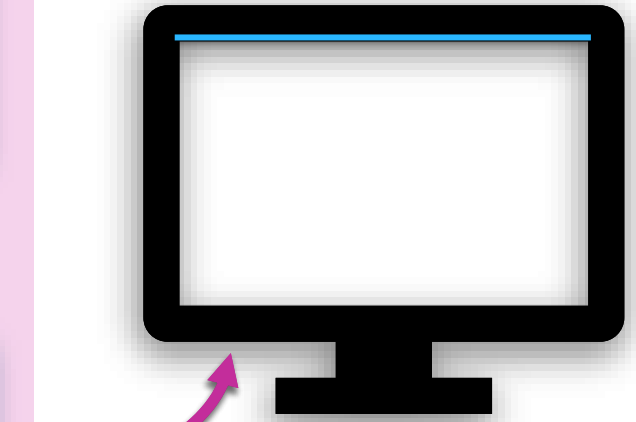
Swap Chain

available images:



to be presented:

presented image:



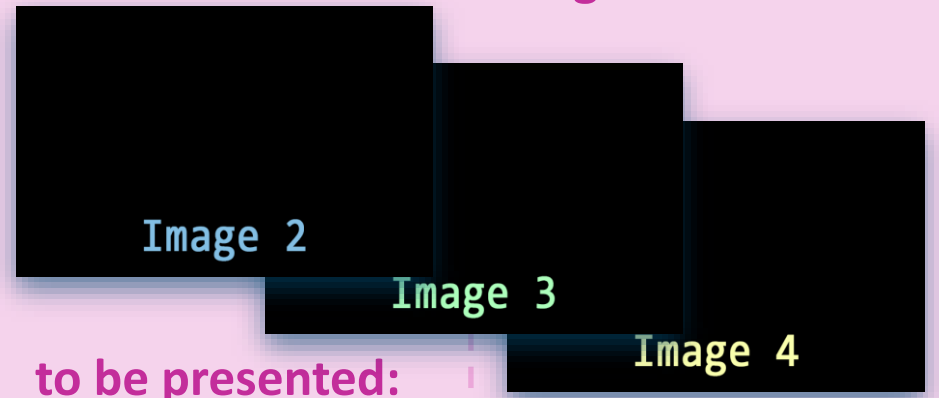
The Swap Chain With a Fast CPU

Application/Render Loop

```
while (true) {

  acquireNextImage();



  draw();



  present();

}
```

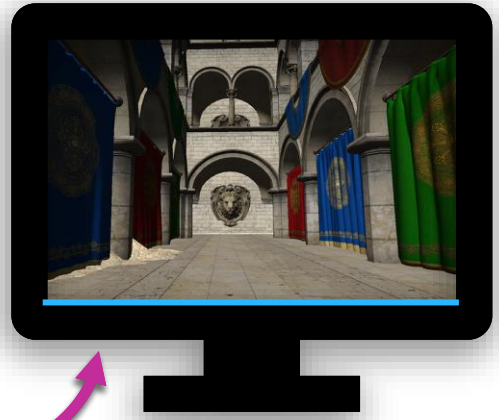
Swap Chain

available images:



to be presented:

presented image:



Introduction to Computer Graphics

186.832, 2021W, 3.0 ECTS

Thank you for your attention!

Johannes Unterguggenberger

Institute of Visual Computing & Human-Centered Technology

TU Wien, Austria