VkPhysicalDevice, VkExtensionProperties, VkSurfaceCapabilitiesKHR, VkSurfaceFormatKHR, and VkPresentModeKHR

```
VkResult vkEnumeratePhysicalDevices(
    VkInstance instance,                     <- VkInstance
    uint32_t* pPhysicalDeviceCount,
    VkPhysicalDevice* pPhysicalDevices       -> VkPhysicalDevice
)
```

```
VkResult vkEnumerateDeviceExtensionProperties(
    VkPhysicalDevice physicalDevice,         <- VkPhysicalDevice
    const char* pLayerName, // usually nullptr
    uint32_t* pPropertyCount,
    VkExtensionProperties* pProperties       -> VkExtensionProperties
)
```

VkExtensionProperties

```
char extensionName[VK_MAX_EXTENSION_NAME_SIZE] // 256 in vulkan_core.h
uint32_t  specVersion
        VK_KHR_16bit_storage  1
        ...
        VK_KHR_swapchain 70
        VK_KHR_zero_initialize_workgroup_memory  1
        VK_EXT_4444_formats  1
        ...
        VK_EXT_ycbcr_image_arrays  1
        VK_NV_clip_space_w_scaling  1
        ...
        VK_NV_viewport_swizzle  1
        VK_NVX_binary_import  1
        VK_NVX_image_view_handle  2
        VK_NVX_multiview_per_view_attributes  1
```

```
VkResult vkGetPhysicalDeviceSurfaceCapabilitiesKHR(
    VkPhysicalDevice          physicalDevice,       <- VkPhysicalDevice
    VkSurfaceKHR              surface,              <- VkSurfaceKHR
    VkSurfaceCapabilitiesKHR* pSurfaceCapabilities  -> VkSurfaceCapabilitiesKHR
)
```

VkSurfaceCapabilitiesKHR

```
uint32_t minImageCount; // 2
uint32_t maxImageCount; // 8
VkExtent2D currentExtent; // (800, 600) depends on the current window
VkExtent2D minImageExtent; // (800, 600) depends on the current window
VkExtent2D maxImageExtent; // (800, 600) depends on the current window
uint32_t maxImageArrayLayers;  // 1
VkSurfaceTransformFlagsKHR supportedTransforms; // VK_SURFACE_TRANSFORM_IDENTITY_BIT_KHR
VkSurfaceTransformFlagBitsKHR currentTransform; // VK_SURFACE_TRANSFORM_IDENTITY_BIT_KHR
VkCompositeAlphaFlagsKHR supportedCompositeAlpha; // VK_COMPOSITE_ALPHA_OPAQUE_BIT_KHR
VkImageUsageFlags supportedUsageFlags; // VK_IMAGE_USAGE_TRANSFER_SRC_BIT
                                       // VK_IMAGE_USAGE_TRANSFER_DST_BIT
                                       // VK_IMAGE_USAGE_SAMPLED_BIT
                                       // VK_IMAGE_USAGE_STORAGE_BIT
                                       // VK_IMAGE_USAGE_COLOR_ATTACHMENT_BIT
                                       // VK_IMAGE_USAGE_INPUT_ATTACHMENT_BIT
```

```
VkResult vkGetPhysicalDeviceSurfaceFormatsKHR(
    VkPhysicalDevice    physicalDevice,     <- VkPhysicalDevice
    VkSurfaceKHR        surface,            <- VkSurfaceKHR
    uint32_t*           pSurfaceFormatCount,
    VkSurfaceFormatKHR* pSurfaceFormats);   -> VkSurfaceFormatKHR
```

VkSurfaceFormatKHR

```
VkFormat        format;
    // VK_FORMAT_B8G8R8A8_UNORM = 44
    // VK_FORMAT_B8G8R8A8_SRGB = 50,
VkColorSpaceKHR colorSpace;
    // VK_COLOR_SPACE_SRGB_NONLINEAR_KHR = 0,
```

```
VkResult vkGetPhysicalDeviceSurfacePresentModesKHR(
    VkPhysicalDevice  physicalDevice,   <- VkPhysicalDevice
    VkSurfaceKHR      surface,          <- VkSurfaceKHR
    uint32_t*         pPresentModeCount,
    VkPresentModeKHR* pPresentModes     -> typedef enum VkPresentModeKHR
        // VK_PRESENT_MODE_IMMEDIATE_KHR = 0,
        // (VK_PRESENT_MODE_MAILBOX_KHR = 1,)
        // VK_PRESENT_MODE_FIFO_KHR = 2,
        // VK_PRESENT_MODE_FIFO_RELAXED_KHR = 3,
);
```

typedef enum VkPresentModeKHR {

```
  VK_PRESENT_MODE_IMMEDIATE_KHR = 0,
  VK_PRESENT_MODE_MAILBOX_KHR = 1,
  VK_PRESENT_MODE_FIFO_KHR = 2,
  VK_PRESENT_MODE_FIFO_RELAXED_KHR = 3,
  VK_PRESENT_MODE_SHARED_DEMAND_REFRESH_KHR,
  VK_PRESENT_MODE_SHARED_CONTINUOUS_REFRESH_KHR,
} VkPresentModeKHR;
```