Vulkan基础设施创建

Host
（CPU 和 内存为基础进行的操作）

Instance
（各渲染程序元数据，VulKan版本，引擎版本，debug启用，扩展开启）

- VkInstance
  - VkInstanceCreateInfo
    （创建instance的各类信息， 并且绑定到 VkInsatnce中）
    - pApplicationInfo
      （传入VkApplicationInfo的指针）
      - sType 设置类型 — VK_STRUCTURE_TYPE_APPLICATION_INFO
      - pApplicationName 设置application指针 — const char*
      - applicationVersion 设置application版本 — VK_MAKE_VERSION(x, x, x)
      - pEngineName 指向包含引擎名称的字符串的指针 — const char*
      - engineVersion 描述 Vulkan 设备所使用的图形引擎的版本号 — VK_MAKE_VERSION(x, x, x)
      - apiVersion api版本 — VK_MAKE_API_VERSION(x,x,x,x)
    - Extensions 扩展
      - enabledExtensionCount 要开启的扩展计数器
      - ppEnabledExtensionNames 代表所有要启用字符串数组
    - sType 设置类型 — VK_STRUCTURE_TYPE_INSTANCE_CREATE_INFO
    - Layer 分为 验证层（Validation Layers）和 性能层（Performance Layers）
      - enabledLayerCount 要开启的Layer计数器
      - ppEnabledLayerNames 代表所有要启用字符串数组

```
vkCreateInstance(VkInstanceCreateInfo*, nullptr, VkInstance*);
```

Surface
（连接操作系统相关的窗口（可以是windows.h 或者 glfw）与vulkan的图像组件）

- VkSurfaceKHR

```
glfwCreateWindowSurface(VkInstance, GLFWwindow*, nullptr, VkSurfaceKHR*);
```

LogicalDevic
（申请一个资源后，需要让逻辑设备Host操作设备）

- 挑选一张符合的显卡

```
vkEnumeratePhysicalDevices(VkInstance, uint32_t*, VkPhysicalDevice 的数组地址);
//获取设备的所有显卡
```

```
vkGetPhysicalDeviceProperties(VkPhysicalDevice, VkPhysicalDeviceProperties*);
//获取显卡类型
vkGetPhysicalDeviceFeatures(VkPhysicalDevice, VkPhysicalDeviceFeatures*);
//获取显卡的支持属性
```

- VkPhysicalDevice 显卡
  - VkPhysicalDeviceProperties 显卡类型
  - VkPhysicalDeviceFeatures 显卡属性
- VkQueue 分为渲染队列和输出队列
  - VkDeviceQueueCreateInfo 队列信息
    - sType 类型设置 — VK_STRUCTURE_TYPE_DEVICE_QUEUE_CREATE_INFO
    - queueFamilyIndex 队列族的下标位置 — uint32_t
    - queueCount 队列的数量 — uint32_t
    - pQueuePriorities 队列优先级 — float
  - VkQueueFamilyProperties 队列族属性
- VkDevice 逻辑设备的对象
  - VkDeviceCreateInfo Device的信息
    - sType 类型设置 — VK_STRUCTURE_TYPE_DEVICE_CREATE_INFO
    - pQueueCreateInfos 代表队列信息的数组 VkDeviceQueueCreateInfo的数组
    - queueCreateInfoCount 队列信息个数的计数
    - pEnabledFeatures 要开启的特点 — VkPhysicalDeviceFeatures
    - enabledExtensionCount 显卡开启的扩展计数器 — uint32_t
    - ppEnabledExtensionNames 要开启的扩展的名字的字符串数组

```
vkCreateDevice(VkPhysicalDevice, VkDeviceQueueCreateInfo*, nullptr, VkDevice*)
```

- uint32_t mGraphicQueueFamily 渲染队列族的下标
- uint32_t mPresentQueueFamily 输出队列族的下标

判断队列的属性信息，来编排下标

```
vkGetPhysicalDeviceQueueFamilyProperties(VkDevice, &queueFamilyCount, null(ptr);
//获得队列的数量
vkGetPhysicalDeviceQueueFamilyProperties(VkDevice, &queueFamilyCount, VkQueueFamilyProperties的数组地址);
//给队列添加数据
```

SwapChain
（交换链，不断把渲染好的 buffer push进队列，对队列头部的buffer输出到屏幕）

- VkSwapchainKHR
  - VkSwapchainCreateInfoKHR
    - sType 设定类型 — VK_STRUCTURE_TYPE_SWAPCHAIN_CREATE_INFO_KHR
    - surface 设定Surface — VkSurfaceKHR
    - minImageCount 设定最小图片数量 — uint32_t
    - imageFormat 设定图片的格式 — VkFormat
    - imageColorSpace 设定图片的颜色空间 — VkColorSpaceKHR
    - imageExtent 设定图片的扩展 — VkExtent2D
    - imageArrayLayers 设定图片层数 — uint32_t
    - imageUsage 图片的使用方式
      - VK_IMAGE_USAGE_TRANSFER_SRC_BIT 图像可以作为传输操作的源图像。
      - VK_IMAGE_USAGE_TRANSFER_DST_BIT 图像可以作为传输操作的目标图像。
      - VK_IMAGE_USAGE_SAMPLED_BIT 图像可以作为采样器的输入。
      - VK_IMAGE_USAGE_STORAGE_BIT 图像可以作为存储图像进行存储操作。
      - VK_IMAGE_USAGE_COLOR_ATTACHMENT_BIT 图像可以作为颜色附件进行颜色渲染操作。
      - VK_IMAGE_USAGE_DEPTH_STENCIL_ATTACHMENT_BIT 图像可以作为深度/模板附件进行深度/模板渲染操作。
      - VK_IMAGE_USAGE_INPUT_ATTACHMENT_BIT 图像可以作为输入附件进行后期处理等操作。
    - imageSharingMode 图片的共享模式
      - VK_SHARING_MODE_EXCLUSIVE 图像只能在一个队列族中使用，不支持在多个队列族之间共享。
      - VK_SHARING_MODE_CONCURRENT 图像可以在多个队列族之间共享，适用于需要在不同队列族之间进行并发操作的情况。
    - queueFamilyIndexCount 队列族的下标的数量 — uint32_t
    - pQueueFamilyIndices 指定队列族索引的指针 — uint32_t*
    - preTransform 指定当前变换模式
    - compositeAlpha 图像之间的透明度效果处理
      - VK_COMPOSITE_ALPHA_OPAQUE_BIT_KHR 交换链图像的 alpha 通道完全不透明，不支持透明度效果。
      - VK_COMPOSITE_ALPHA_PRE_MULTIPLIED_BIT_KHR 交换链图像的 alpha 通道已经预乘了颜色值，适用于预乘了 alpha 通道的图像。
      - VK_COMPOSITE_ALPHA_POST_MULTIPLIED_BIT_KHR 交换链图像的 alpha 通道在颜色值之后进行组合，适用于后乘了 alpha 通道的图像。
      - VK_COMPOSITE_ALPHA_INHERIT_BIT_KHR 交换链图像的 alpha 通道继承自窗口系统，适用于从窗口系统继承 alpha 通道的情况。
    - presentMode 显示模式
    - clipped 剪裁是否开启 — VK_TREU / VK_FALSE

```
vkCreateSwapchainKHR(VkDivice, VkSwapchainCreateInfoKHR*, nullptr, VkSwapchainKHR* );
```

- VkFormat 表示图像或缓冲区数据
- VkExtent2D 表示二维图像或区域的尺寸
  - width
  - heigth
- VkImage 图像对象的数据结构 在这里使用了这个对象的数组
  - imageType 指定图像的类型，如 1D、2D、3D。
  - format 指定图像的像素格式，如 RGBA8_UNORM、DEPTH_STENCIL等。
  - extent 指定图像的宽度、高度和深度。
  - mipLevels 指定图像的 mip 层级数量。
  - arrayLayers 指定图像的数组层级数量。
  - samples 指定图像的采样数，用于多重采样。
  - usage 指定图像的用途，如颜色附件、深度附件、纹理等。
  - initialLayout 指定图像的初始布局，如 UNDEFINED、PREINITIALIZED 等。
  - tiling 指定图像的存储布局，如 OPTIMAL、LINEAR 等。

```
vkGetSwapchainImagesKHR(VkDevice, VkSwapchainKHR, uint32_t*, nullptr);
vkGetSwapchainImagesKHR(VkDevice, VkSwapchainKHR, uint32_t*, VkImage的数组地址)
```

- VkImageView 描述了如何访问 VkImage 中的数据 在这里使用了这个对象的数组
  - VkImageViewCreateInfo VkImageView的信息
    - image 指向要创建视图的图像对象的句柄。 — VkImage
    - viewType 指定视图的类型，
    - format 指定视图的格式，如 RGBA8_UNORM、DEPTH_STENCIL等。 — VkFormat
    - components 指定对图像中的每个颜色通道进行的映射和重映射操作。 — VkImageAspectFlags
      - VK_IMAGE_ASPECT_COLOR_BIT 指定图像包含颜色数据。
      - VK_IMAGE_ASPECT_DEPTH_BIT 指定图像包含深度数据。
      - VK_IMAGE_ASPECT_STENCIL_BIT 指定图像包含模板数据。
      - VK_IMAGE_ASPECT_METADATA_BIT 指定图像包含元数据。
    - subresourceRange 指定视图所涵盖的图像的范围，包括 mip 级别、数组层级和图像的各个方面。
      - aspectMask 指定图像视图的方面 — VK_IMAGE_ASPECT_COLOR_BIT、VK_IMAGE_ASPECT_DEPTH_BIT、VK_IMAGE_ASPECT_STENCIL_BIT
      - baseMipLevel 指定图像的起始 mip 级别。
      - levelCount 指定 mip 级别的数量。
      - baseArrayLayer 指定起始数组层级。
      - layerCount 指定数组层级的数量。

```
vkCreateImageView(VkDevice, VkImageViewCreateInfo*, nullptr, VkImageView*);
```

- SwapChainSupportInfo 一个自定义的结构体
  - VkSurfaceCapabilitiesKHR 对Surface容量的设定
    - minImageCount 表面支持的最小图像数量。
    - maxImageCount 表面支持的最大图像数量。如果为 0，则表示没有限制。
    - currentExtent 表面的当前尺寸（宽度和高度），如果宽度和高度都为特殊值 0xFFFFFFFF，则表示表面的尺寸由应用程序决定。
    - minImageExtent 表面支持的最小图像尺寸。
    - maxImageExtent 表面支持的最大图像尺寸。
    - maxImageArrayLayers 表面支持的最大图像数组层数量。
    - supportedTransforms 表面支持的变换方式（例如旋转、翻转等）。
    - currentTransform 表面的当前变换方式。
    - supportedCompositeAlpha 表面支持的 Alpha 混合方式。
    - supportedUsageFlags 表面支持的图像使用标志位。

```
vkGetPhysicalDeviceSurfaceCapabilitiesKHR( VkPhysicalDevice, VkSurfaceKHR, VkSurfaceCapabilitiesKHR*);
```

  - VkSurfaceFormatKHR 对Surface的图像支持格式的定义 在该结构体用的是该类型的数组 — 传入数组
  - VkPresentModeKHR 描述交换链支持的呈现模式的枚举类型 在该结构体用的是该类型的数组 — 传入数组

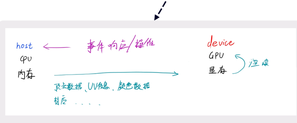
```
vkGetPhysicalDeviceSurfaceFormatsKHR(VkPhysicalDevice, VkSurfaceKHR, uint32_t* , nullptr );
vkGetPhysicalDeviceSurfaceFormatsKHR(VkPhysicalDevice, VkSurfaceKHR, uint32_t* , VkSurfaceFormatKHR的数组的地址 );
```

```
vkGetPhysicalDeviceSurfacePresentModesKHR(VkPhysicalDevice, VkSurfaceKHR, uint32_t* , nullptr);
vkGetPhysicalDeviceSurfacePresentModesKHR(VkPhysicalDevice, VkSurfaceKHR, uint32_t* , VkPresentModeKHR数组的地址);
```

选择一个合适的格式

选择一个合适的呈现模式

判断



Host 和 Device端 之间的操作

Device
（GPU与显存的操作）

- PhysicalDevic
  （表示使用哪一张显卡作为渲染）
- Buffer1
  Buffer2
  Buffer3
  Buffer4
  Buffer5
  ——

Presented with xmind