

VkDescriptorSet

VkDescriptorSetAllocateInfo

```
sType = VK_STRUCTURE_TYPE_DESCRIPTOR_SET_ALLOCATE_INFO;  
pNext = nullptr;  
descriptorPool; ←  
descriptorSetCount;  
pSetLayouts; ←
```

VkDescriptorPool

VkDescriptorSetLayout

```
VkResult vkAllocateDescriptorSets(  
    VkDevice device, ←  
    const VkDescriptorSetAllocateInfo* pAllocateInfo,  
    VkDescriptorSet* pDescriptorSets  
);
```

VkDevice

```
VkResult vkFreeDescriptorSets(  
    VkDevice device, ←  
    VkDescriptorPool descriptorPool, ←  
    uint32_t descriptorSetCount,  
    const VkDescriptorSet* pDescriptorSets ←  
);
```

VkDevice

VkDescriptorPool

VkDescriptorSetLayout

typedef struct VkDescriptorBufferInfo {

```
VkBuffer buffer; ←  
VkDeviceSize offset;  
VkDeviceSize range;  
} VkDescriptorBufferInfo;
```

VkBuffer

typedef struct VkDescriptorImageInfo {

```
VkSampler sampler; ←  
VkImageView imageView; ←  
VkImageLayout imageLayout; ←  
} VkDescriptorImageInfo;
```

VkSampler

VkImageView

typedef enum VkImageLayout {

```
VK_IMAGE_LAYOUT_UNDEFINED = 0,  
VK_IMAGE_LAYOUT_GENERAL = 1,  
VK_IMAGE_LAYOUT_COLOR_ATTACHMENT_OPTIMAL = 2,  
VK_IMAGE_LAYOUT_DEPTH_STENCIL_ATTACHMENT_OPTIMAL = 3,  
VK_IMAGE_LAYOUT_DEPTH_STENCIL_READ_ONLY_OPTIMAL = 4,  
VK_IMAGE_LAYOUT_SHADER_READ_ONLY_OPTIMAL = 5,  
VK_IMAGE_LAYOUT_TRANSFER_SRC_OPTIMAL = 6,  
VK_IMAGE_LAYOUT_TRANSFER_DST_OPTIMAL = 7,  
VK_IMAGE_LAYOUT_PREINITIALIZED = 8,  
...  
} VkImageLayout;
```

VkWriteDescriptorSet

```
sType = VK_STRUCTURE_TYPE_WRITE_DESCRIPTOR_SET;  
pNext = nullptr;  
dstSet; // destination to write  
dstBinding;  
// This must match the binding number in the shaders.  
// Ex.  
// layout(binding = 0) uniform UniformBufferObject{...}ubo;  
// layout(binding = 1) uniform sampler2D texSampler;  
dstArrayElement;  
descriptorCount;  
descriptorType; ←  
pImageInfo;  
pBufferInfo;  
pTexelBufferView = nullptr;
```

typedef enum VkDescriptorType {

```
VK_DESCRIPTOR_TYPE_SAMPLER = 0,  
VK_DESCRIPTOR_TYPE_COMBINED_IMAGE_SAMPLER = 1,  
VK_DESCRIPTOR_TYPE_SAMPLED_IMAGE = 2,  
VK_DESCRIPTOR_TYPE_STORAGE_IMAGE = 3,  
VK_DESCRIPTOR_TYPE_UNIFORM_TEXEL_BUFFER = 4,  
VK_DESCRIPTOR_TYPE_STORAGE_TEXEL_BUFFER = 5,  
VK_DESCRIPTOR_TYPE_UNIFORM_BUFFER = 6,  
VK_DESCRIPTOR_TYPE_STORAGE_BUFFER = 7,  
VK_DESCRIPTOR_TYPE_UNIFORM_BUFFER_DYNAMIC = 8,  
VK_DESCRIPTOR_TYPE_STORAGE_BUFFER_DYNAMIC = 9,  
VK_DESCRIPTOR_TYPE_INPUT_ATTACHMENT = 10,  
...  
} VkDescriptorType;
```

VkCopyDescriptorSet

```
sType = VK_STRUCTURE_TYPE_COPY_DESCRIPTOR_SET;  
pNext = nullptr;  
srcSet;  
srcBinding;  
srcArrayElement;  
dstSet;  
dstBinding;  
dstArrayElement;  
descriptorCount;
```

```
void vkUpdateDescriptorSets(  
    VkDevice device, ←  
    uint32_t descriptorWriteCount,  
    const VkWriteDescriptorSet* pDescriptorWrites,  
    uint32_t descriptorCopyCount,  
    const VkCopyDescriptorSet* pDescriptorCopies  
);
```

VkDevice