VkCommandPool

```
typedef enum VkCommandPoolCreateFlagBits {
    VK_COMMAND_POOL_CREATE_TRANSIENT_BIT = 0x00000001,
    VK_COMMAND_POOL_CREATE_RESET_COMMAND_BUFFER_BIT = 0x00000002,
  // Provided by VK_VERSION_1_1
    VK_COMMAND_POOL_CREATE_PROTECTED_BIT = 0x00000004,
} VkCommandPoolCreateFlagBits;
```

```
VkCommandPoolCreateInfo
sType = VK_STRUCTURE_TYPE_COMMAND_POOL_CREATE_INFO;
pNext = nullptr;
flags = VK_COMMAND_POOL_CREATE_RESET_COMMAND_BUFFER_BIT;
queueFamilyIndex;
```

```
VkResult vkCreateCommandPool(
    VkDevice                        device,          <- VkDevice
    const VkCommandPoolCreateInfo*  pCreateInfo,
    const VkAllocationCallbacks*    pAllocator,
    VkCommandPool*                  pCommandPool     -> VkCommandPool
);
```

```
void vkDestroyCommandPool(
    VkDevice                    device,          <- VkDevice
    VkCommandPool               commandPool,     <- VkCommandPool
    const VkAllocationCallbacks* pAllocator
);
```

VkCommandBuffer

```
typedef enum VkCommandBufferLevel {
    VK_COMMAND_BUFFER_LEVEL_PRIMARY = 0,
    VK_COMMAND_BUFFER_LEVEL_SECONDARY = 1,
} VkCommandBufferLevel;
```

```
VkCommandBufferAllocateInfo
sType = VK_STRUCTURE_TYPE_COMMAND_BUFFER_ALLOCATE_INFO;
pNext = nullptr;
commandPool;
level;
commandBufferCount; // num buffers to allocate
```

```
VkResult vkAllocateCommandBuffers(
    VkDevice                            device,          <- VkDevice
    const VkCommandBufferAllocateInfo*  pAllocateInfo,
    VkCommandBuffer*                    pCommandBuffers  -> VkCommandBuffer
);
```

```
void vkFreeCommandBuffers(
    VkDevice                device,          <- VkDevice
    VkCommandPool           commandPool,     <- VkCommandPool
    uint32_t                commandBufferCount,
    const VkCommandBuffer*  pCommandBuffers);
```