VkPipelineShaderStageCreateInfo for VkGraphicsPipelineCreateInfo

```
VkShaderModuleCreateInfo
sType = VK_STRUCTURE_TYPE_SHADER_MODULE_CREATE_INFO;
pNext; // usually nullptr
flags = 0;
codeSize; // in bytes
pCode; // pointer to the compiled SPIR-V byte code
```

```
VkResult vkCreateShaderModule(
    VkDevice                        device,
    const VkShaderModuleCreateInfo* pCreateInfo,
    const VkAllocationCallbacks*    pAllocator,
    VkShaderModule*                 pShaderModule
);
```

VkDevice

VkShaderModule

```
void vkDestroyShaderModule(
    VkDevice                     device,
    VkShaderModule               shaderModule,
    const VkAllocationCallbacks* pAllocator);
```

VkDevice

VkShaderModule

```
typedef enum VkShaderStageFlagBits {
    VK_SHADER_STAGE_VERTEX_BIT = 0x00000001,
    VK_SHADER_STAGE_TESSELLATION_CONTROL_BIT = 0x00000002,
    VK_SHADER_STAGE_TESSELLATION_EVALUATION_BIT = 0x00000004,
    VK_SHADER_STAGE_GEOMETRY_BIT = 0x00000008,
    VK_SHADER_STAGE_FRAGMENT_BIT = 0x00000010,
    VK_SHADER_STAGE_COMPUTE_BIT = 0x00000020,
    VK_SHADER_STAGE_ALL_GRAPHICS = 0x0000001F,
    ...
} VkShaderStageFlagBits;
```

```
VkPipelineShaderStageCreateInfo
sType = VK_STRUCTURE_TYPE_PIPELINE_SHADER_STAGE_CREATE_INFO;
pNext = nullptr;
flags; // usually 0.
stage;
module;
pName = <SHADER_STAGE_NAME>;
pSpecializationInfo; // usually nullptr
```

VkPipelineVertexInputStateCreateInfo for VkGraphicsPipelineCreateInfo

```
typedef enum VkFormat {
    ...
    VK_FORMAT_R32_UINT = 98,
    VK_FORMAT_R32_SINT = 99,
    VK_FORMAT_R32_SFLOAT = 100,
    VK_FORMAT_R32G32_UINT = 101,
    VK_FORMAT_R32G32_SINT = 102,
    VK_FORMAT_R32G32_SFLOAT = 103,
    VK_FORMAT_R32G32B32_UINT = 104,
    VK_FORMAT_R32G32B32_SINT = 105,
    VK_FORMAT_R32G32B32_SFLOAT = 106,
    VK_FORMAT_R32G32B32A32_UINT = 107,
    VK_FORMAT_R32G32B32A32_SINT = 108,
    VK_FORMAT_R32G32B32A32_SFLOAT = 109,
    ...
} VkFormat;
```

```
VkVertexInputBindingDescription
binding; // number/index of the buffer
stride;  // in bytes
VkVertexInputRate    inputRate;
    // VK_VERTEX_INPUT_RATE_VERTEX or
    // VK_VERTEX_INPUT_RATE_INSTANCE
```

```
VkVertexInputAttributeDescription
uint32_t    location; // index in the shader language
uint32_t    binding;  // number/index of the buffer
VkFormat    format;
uint32_t    offset;
```

```
VkPipelineVertexInputStateCreateInfo
sType = VK_STRUCTURE_TYPE_PIPELINE_VERTEX_INPUT_STATE_CREATE_INFO;
pNext = nullptr;
flags = 0;
vertexBindingDescriptionCount;
pVertexBindingDescriptions;
vertexAttributeDescriptionCount;
pVertexAttributeDescriptions;
```