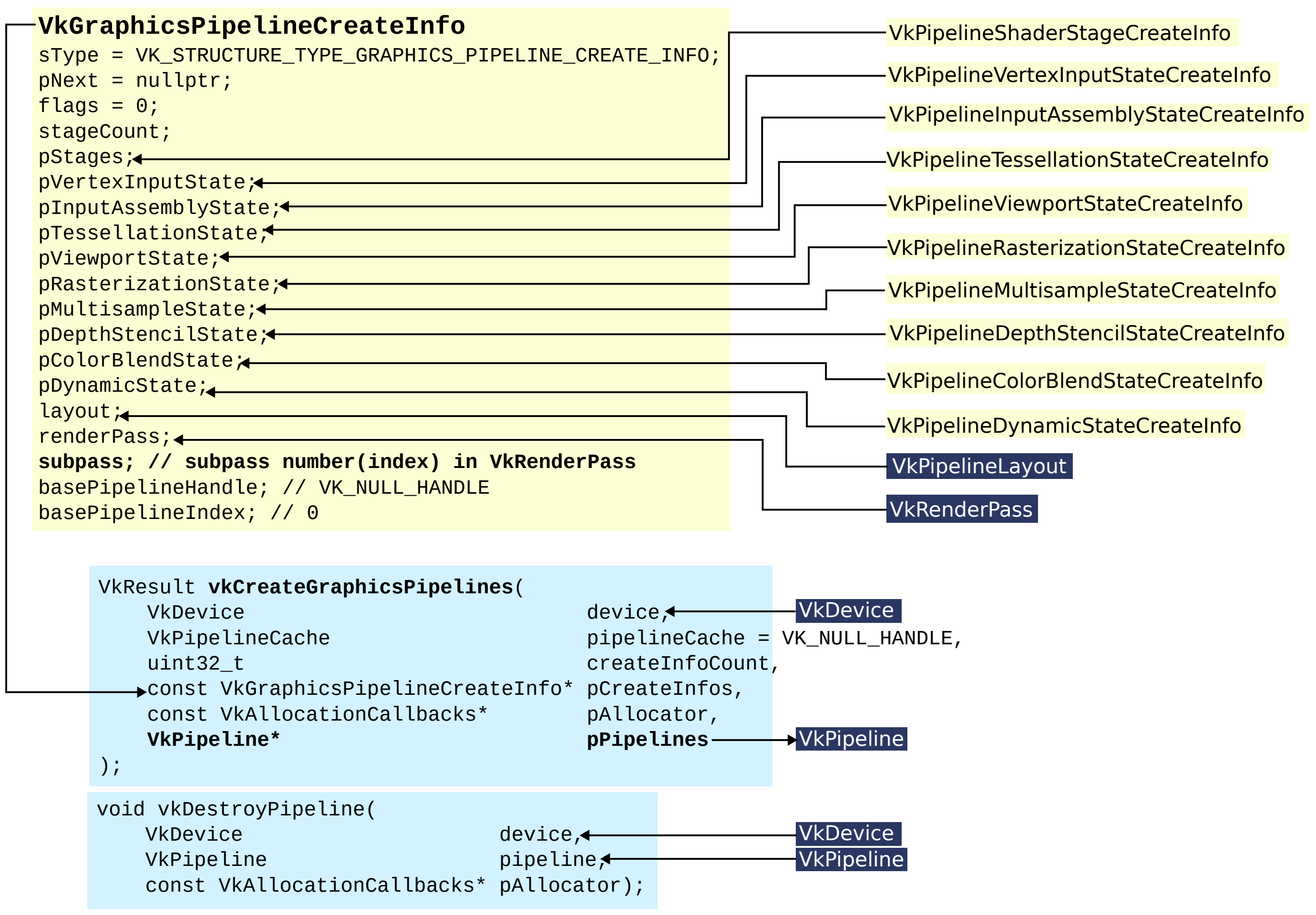
VkPipeline



```
VkGraphicsPipelineCreateInfo
sType = VK_STRUCTURE_TYPE_GRAPHICS_PIPELINE_CREATE_INFO;
pNext = nullptr;
flags = 0;
stageCount;
pStages;
pVertexInputState;
pInputAssemblyState;
pTessellationState;
pViewportState;
pRasterizationState;
pMultisampleState;
pDepthStencilState;
pColorBlendState;
pDynamicState;
layout;
renderPass;
subpass; // subpass number(index) in VkRenderPass
basePipelineHandle; // VK_NULL_HANDLE
basePipelineIndex; // 0
```

```
VkResult vkCreateGraphicsPipelines(
    VkDevice                            device,
    VkPipelineCache                     pipelineCache = VK_NULL_HANDLE,
    uint32_t                            createInfoCount,
    const VkGraphicsPipelineCreateInfo* pCreateInfos,
    const VkAllocationCallbacks*        pAllocator,
    VkPipeline*                         pPipelines
);
```

```
void vkDestroyPipeline(
    VkDevice                     device,
    VkPipeline                   pipeline,
    const VkAllocationCallbacks* pAllocator);
```

VkPipelineShaderStageCreateInfo
VkPipelineVertexInputStateCreateInfo
VkPipelineInputAssemblyStateCreateInfo
VkPipelineTessellationStateCreateInfo
VkPipelineViewportStateCreateInfo
VkPipelineRasterizationStateCreateInfo
VkPipelineMultisampleStateCreateInfo
VkPipelineDepthStencilStateCreateInfo
VkPipelineColorBlendStateCreateInfo
VkPipelineDynamicStateCreateInfo
VkPipelineLayout
VkRenderPass
VkDevice
VkPipeline
VkDevice
VkPipeline