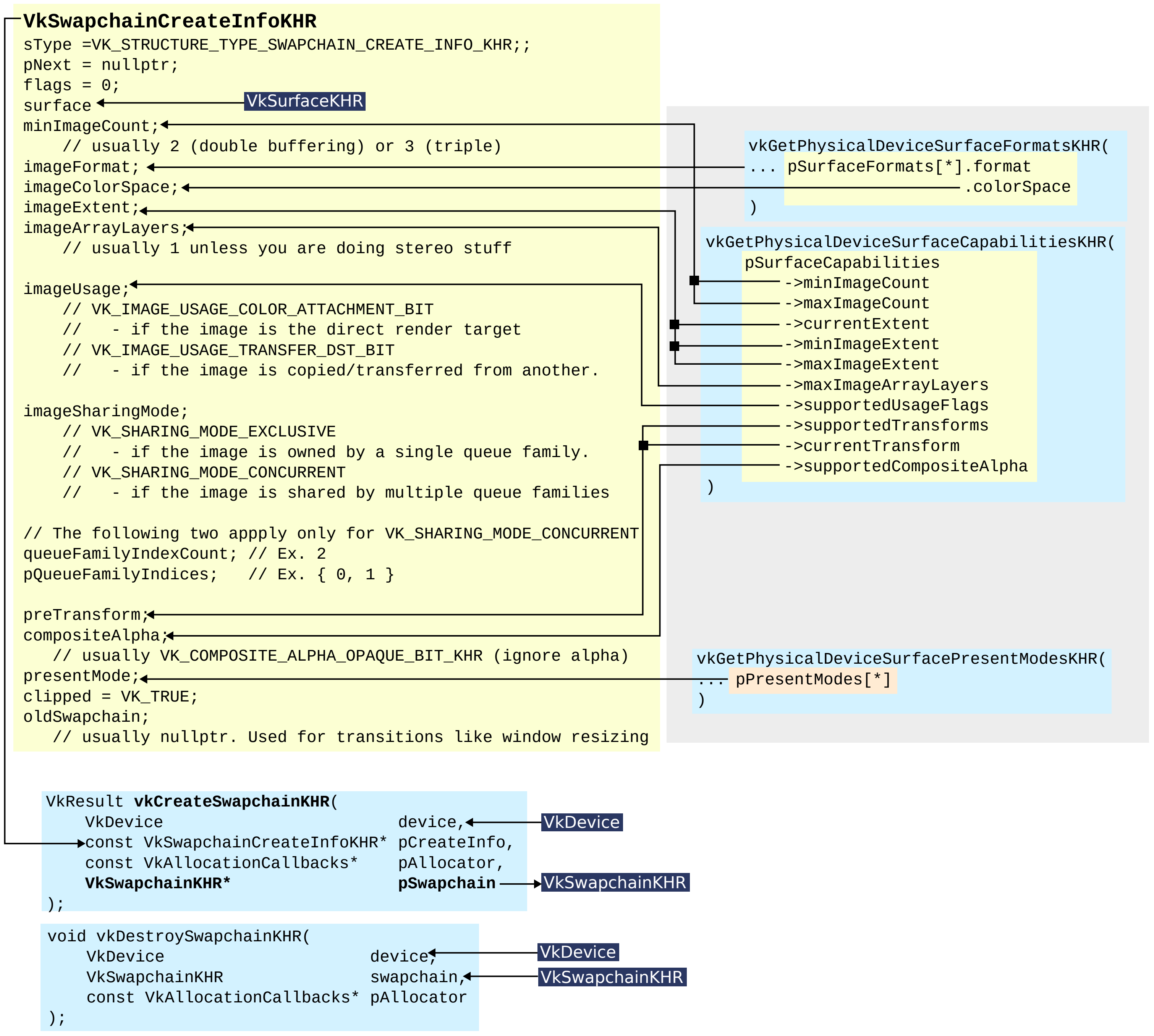
VkSwapchainKHR



```
VkSwapchainCreateInfoKHR
sType =VK_STRUCTURE_TYPE_SWAPCHAIN_CREATE_INFO_KHR;;
pNext = nullptr;
flags = 0;
surface            <- VkSurfaceKHR
minImageCount;
    // usually 2 (double buffering) or 3 (triple)
imageFormat;
imageColorSpace;
imageExtent;
imageArrayLayers;
    // usually 1 unless you are doing stereo stuff

imageUsage;
    // VK_IMAGE_USAGE_COLOR_ATTACHMENT_BIT
    //   - if the image is the direct render target
    // VK_IMAGE_USAGE_TRANSFER_DST_BIT
    //   - if the image is copied/transferred from another.

imageSharingMode;
    // VK_SHARING_MODE_EXCLUSIVE
    //   - if the image is owned by a single queue family.
    // VK_SHARING_MODE_CONCURRENT
    //   - if the image is shared by multiple queue families

// The following two appply only for VK_SHARING_MODE_CONCURRENT
queueFamilyIndexCount; // Ex. 2
pQueueFamilyIndices;   // Ex. { 0, 1 }

preTransform;
compositeAlpha;
   // usually VK_COMPOSITE_ALPHA_OPAQUE_BIT_KHR (ignore alpha)
presentMode;
clipped = VK_TRUE;
oldSwapchain;
   // usually nullptr. Used for transitions like window resizing
```

```
vkGetPhysicalDeviceSurfaceFormatsKHR(
... pSurfaceFormats[*].format
                       .colorSpace
)
```

```
vkGetPhysicalDeviceSurfaceCapabilitiesKHR(
    pSurfaceCapabilities
        ->minImageCount
        ->maxImageCount
        ->currentExtent
        ->minImageExtent
        ->maxImageExtent
        ->maxImageArrayLayers
        ->supportedUsageFlags
        ->supportedTransforms
        ->currentTransform
        ->supportedCompositeAlpha
)
```

```
vkGetPhysicalDeviceSurfacePresentModesKHR(
... pPresentModes[*]
)
```

```
VkResult vkCreateSwapchainKHR(
    VkDevice                         device,      <- VkDevice
    const VkSwapchainCreateInfoKHR*  pCreateInfo,
    const VkAllocationCallbacks*     pAllocator,
    VkSwapchainKHR*                  pSwapchain   -> VkSwapchainKHR
);
```

```
void vkDestroySwapchainKHR(
    VkDevice                     device,     <- VkDevice
    VkSwapchainKHR               swapchain,  <- VkSwapchainKHR
    const VkAllocationCallbacks* pAllocator
);
```