Fun with border colors in Vulkan

A story about VK_EXT_border_color_swizzle

Ricardo Garcia

2022-02-06 FOSDEM

Sampling in Vulkan

Image View (B8G8R8_UNORM)

B	G	R
00010000	11110111	00010110

GLSL

```
vec4 color = texture(sampler, coords);
```

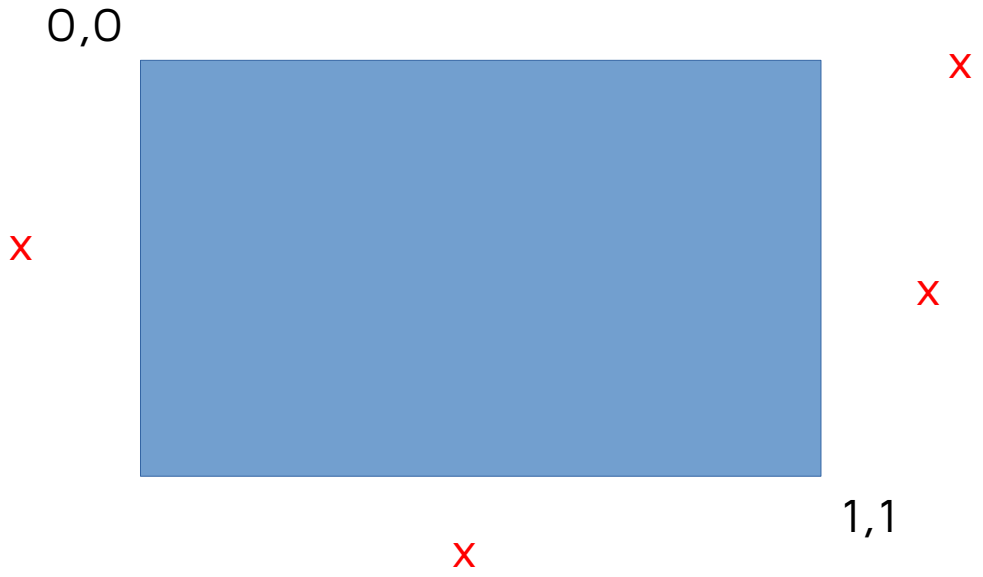
```
color.r = 0.086274509803921 6
color.g = 0.968627450980392 2
color.b = 0.062745098039215 7
color.a = 1.0
```


Normalized Coordinates

GLSL

```
vec4 color = texture(sampler, coords);
```

```
VkSamplerCreateInfo
.addressModeU
.addressModeV
.addressModeW
```



Address Mode

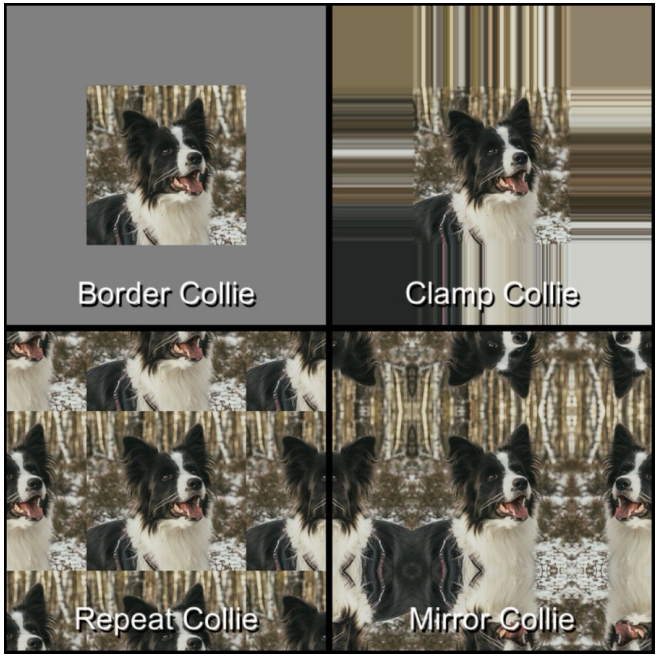


Image source: Aras Pranckevičius, https://twitter.com/aras_p/status/1480199231772237826

Border Color

```
VkSamplerCreateInfo::borderColor

VK_BORDER_COLOR_FLOAT_TRANSPARENT_BLACK
VK_BORDER_COLOR_FLOAT_OPAQUE_BLACK
VK_BORDER_COLOR_FLOAT_OPAQUE_WHITE

// VK_EXT_custom_border_color
VK_BORDER_COLOR_FLOAT_CUSTOM_EXT
```


Image View Swizzle

Image View (B8G8R8_UNORM)

B	G	R
00010000	11110111	00010110

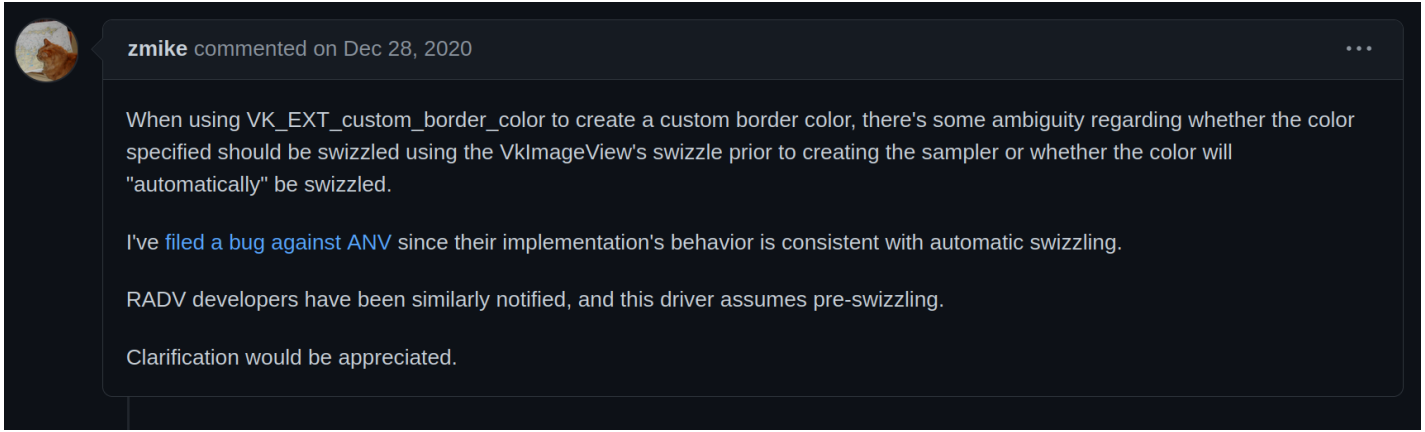
GLSL

```
vec4 color = texture(sampler, coords);
```

```
// Component reordering,
// replacement or duplication.
VkImageViewCreateInfo::components
{ VkComponentSwizzle r, g, b, a; }

color.r = 0.086274509803921 6
color.g = 0.968627450980392 2
color.b = 0.062745098039215 7
color.a = 1.0
```


Border Color and Swizzle



zmike commented on Dec 28, 2020

When using VK_EXT_custom_border_color to create a custom border color, there's some ambiguity regarding whether the color specified should be swizzled using the VkImageView's swizzle prior to creating the sampler or whether the color will "automatically" be swizzled.

I've filed a bug against ANV since their implementation's behavior is consistent with automatic swizzling.

RADV developers have been similarly notified, and this driver assumes pre-swizzling.

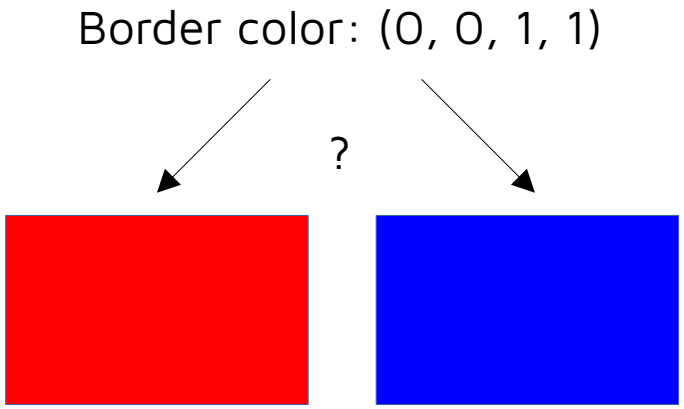
Clarification would be appreciated.

https://github.com/KhronosGroup/Vulkan-Docs/issues/1421

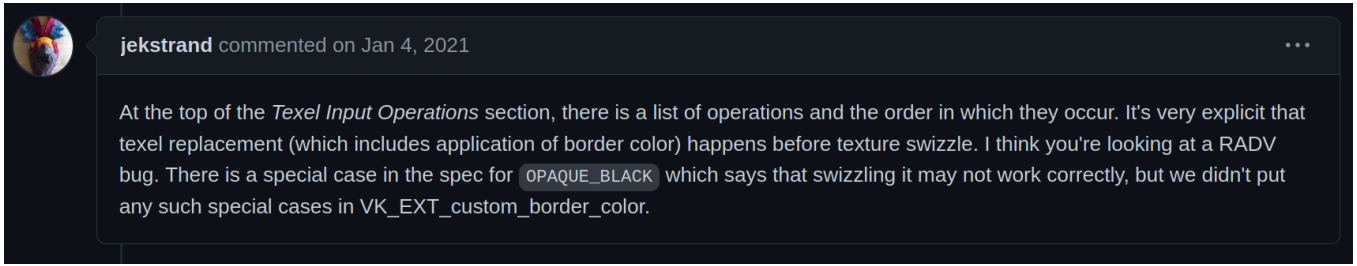
Border Color and Swizzle

```
VkImageViewCreateInfo::components
{
  VK_COMPONENT_SWIZZLE_B,
  VK_COMPONENT_SWIZZLE_ZERO,
  VK_COMPONENT_SWIZZLE_G,
  VK_COMPONENT_SWIZZLE_IDENTITY,
}

VkSamplerCreateInfo::addressMode*
  VK_SAMPLER_ADDRESS_MODE_CLAMP_TO_BORDER
```



Border Color and Swizzle



jekstrand commented on Jan 4, 2021

At the top of the *Texel Input Operations* section, there is a list of operations and the order in which they occur. It's very explicit that texel replacement (which includes application of border color) happens before texture swizzle. I think you're looking at a RADV bug. There is a special case in the spec for `OPAQUE_BLACK` which says that swizzling it may not work correctly, but we didn't put any such special cases in VK_EXT_custom_border_color.

https://github.com/KhronosGroup/Vulkan-Docs/issues/1421

Texel Input Operations

1)Coordinate conversion
2)Coordinate validation (will Texel Replacement happen?)
3)Reading texel from image memory (and rearrange components)
4)Format conversion (e.g. UNORM or sRGB)
5)Texel Replacement (border colors are replaced here)
6)Expansion to RGBA
7)Component swizzle

Coordinate Conversion

Converting normalized coordinates to integer texel coordinates.
Clamping and modifying values depending on the addressing mode.

Coordinate Validation

Deciding if texel replacement will take place.
Related to border colors as well as robustness features.

Reading Texel from Image

Actually getting the value from image memory.
Implies reordering the existing components (R, RG, RGB, RGBA) to standard RGBA order (e.g. BGR to RGB).

Format Conversion

Conversion of integer values to float.
Applying sRGB curve if needed.

Texel in RGB order, UNORM

R	G	B
00010110	11110111	00010000

```
color.r = 0.086274509803921 6
color.g = 0.968627450980392 2
color.b = 0.062745098039215 7
```


Texel Replacement

Includes taking the border color and cutting it short so it only has the components present in the original image, to act “as if” the border color was actually part of the image.

Example: BGR_UNORM format

```
Border color:    (0.0, 0.0, 1.0, 0.5)
Components used: (0.0, 0.0, 1.0)
```


Expansion to RGBA

Expanding the texel color (which may come from the border) to the 4 standard components: RGBA. Missing color components are filled with zeros and if the alpha is missing it's filled with one.

Example: BGR_UNORM format

```
Border color:       (0.0, 0.0, 1.0, 0.5)
Components used:    (0.0, 0.0, 1.0)
Expanded to RGBA:   (0.0, 0.0, 1.0, 1.0)
```


Component Swizzle

Apply component swizzle from image view.

Example: BGR_UNORM format. Swizzle **B**, **Zero**, **One**, **Identity**.

```
Border color:       (0.0, 0.0, 1.0, 0.5)
Components used:    (0.0, 0.0, 1.0)
Expanded to RGBA:   (0.0, 0.0, 1.0, 1.0)
Swizzle applied:    (1.0, 0.0, 1.0, 1.0)
```


VK_EXT_custom_border_color

Already out. No restrictions on non-identity swizzle. What to do?

Option A) Vendors should fix their implementations!
Can they do this? What if not possible/practical?

Option B) Backpedal a bit, make behavior undefined unless some other feature is present. Allow implementations to ship custom border colors.

VK_EXT_border_color_swizzle

Allows indicating the implementation does not support rearranging color components with problematic colors, including custom ones.

`VkPhysicalDeviceBorderColorSwizzleFeaturesEXT::borderColorSwizzle`

Allows indicating the implementation supports that, but applications need to provide the color swizzle also when creating the sampler.

`...BorderColorSwizzleFeaturesEXT::borderColorSwizzleFromImage`
`VkSamplerBorderColorComponentMappingCreateInfoEXT`

Drawback: lowers the bar for applications wanting to use a single code path.

Questions?

We're hiring!

https://www.igalia.com/jobs/


igalia