YUV sampling in Vulkan VK_KHR_sampler_ycbcr_conversion

Lina Liu Nov. 30 2022

What is YUV?

A color model

1. What is YUV?
    1. is describe colorspace that are encoded using YCbCr
    2. encodes color image or video taking human perception into account
    3. allows reduced bandwidth for chrominance components, compared to RGB
    4. Y — Luminance plane, can be seen as the image as grayscale.
        1. Physical linear-space brightness
    5. U — blue projection, *chroma* planes, basically the colours
    6. V — red projection, *chroma* planes, basically the colours
    7. All the YUV formats have these three planes, and differ by the different orderings of them.
    8. About YUV format:
        1. https://gist.github.com/Jim-Bar/3cbba684a71d1a9d468a6711a6eddbeb
        2.

Why YUV is some kind of complicated?

Video compression

2. Why YUV is some kind of complicated?
    1. Planar: Each color component is packed in different 2D images
    2. Luma: Y refers to luminance
    3. UV(CbCr) refers to chrominance(color)
    4. Downsampled chroma. Less bandwidth on color is an easy way to save space.
    5. Different various of YUV format…
        1. How many planes? 2 or 3
        2. Which color component comes first?
        3. How many bit per component? 8-bit or 10-bit?
        4. How much is chroma downsampled? 2x?
        5. Where is the telex center for the chroma samples?
        6. What is exact color space conversion matrix from YUV to RGB?
        7. How is chroma reconstructed to full resolution?

Dealing with YUV without fancy extensions

How many formats you need to deal with

3. Shader variants may quickly get out hand if too many formats

```
layout(binding = 0) uniform TexLuma;
layout(binding = 1) uniform TexCb;
layout(binding = 2) uniform TexCr;

layout(location = 0) out vec3 FragColor;
layout(location = 0) in vec2 TexCoord;

const mat3 yuv_to_rgb_matrix = mat3(....);

void main()
{
    float Luma = textureLod(TexLuma, TexCoord, 0.0).x;
    float Cb = textureLod(TexCb, TexCoord, 0.0).x; // For mid-point chroma
    float Cr = textureLod(TexCr, TexCoord, 0.0).x;
    vec3 yuv = vec3(Luma, Cb, Cr);
    // Possibly expand range here if using TV YUV range and not PC YUV range.
    yuv = rescale_yuv(yuv);
    FragColor = yuv_to_rgb_matrix * yuv;
}
```

VK_KHR_sampler_ycbcr_conversion

VK_FORMAT_G8_B8_R8_3PLANE_420_UNORM

4. Vulkan add new texture format for YUV:
    1. VK_KHR_sampler_ycbcr_conversion
    2. VK_FORMAT_G8_B8_R8_3PLANE_420_UNORM
        1. 420 here means the second and third component are half resolution
        2. GPU can sample 3 samples at once, meaning we will put a lot less stress on the GPU texturing unit.
            1. Means a lot for lower-end mobile devices.

YUV sampling in Vulkan

The image aspects

1. We need to refer to each plane separately when copying data in and out of the texture.
    1. Use VK_IMAGE_ASPECT_PLANE_{0,1,2}_BIT
    2. When copying to/from plane 1 and 2 in YUV420p, the resolutions of those planes are halved.
3. VK_IMAGE_ASPECT_COLOR_BIT refers to the whole “GBR” as a whole, it’s only useful when sampling the image?

YUV sampling steps in Vulkan

5. YUV sampling steps in Vulkan
    1. Create R8_UNORM image with VK_IMAGE_CREATE_ALIAS_BIT
        1. Be aware of alignment requirement
        2. Using standalone allocations per plane, or bumping alignment to something like 64k works around that
    2. When we create planar texture, we specify DISJOINT_BIT and ALIAS_BIT.
        1. For disjoint, it means we need to query allocation requirements and bind memory separately for each plane.
            1. Use vkGetImageMemoryRequirement2
            2. Use vkBindImageMemory2
        2. May bind the same memory we used for our separate textures.
3. Setting up a sampler conversion object
4. Passing along to VkImageView and VkSampler
5. Shader implement

YUV sampling in Vulkan

Setting up a sampler conversion object