Layout Transitions

Vulkan layout transitions

相比于dx11和opengl，vulkan和dx12 多了资源转换屏障的操作

if we are writing to a resource, say a texture, we will set the texture state to a render target state; when we need to read the texture, we will change its state to a shader resource state.

大概意思就是对于资源的使用例如Image 这种需要定义其状态，比如当前是可读还是可写，或者是其他类型，避免同时被两个进程读写，以预防**resource hazard** 状态

同时，从官方给出来的数据来看，资源transitions了之后，而不是没指定的VK_IMAGE_LAYOUT_UNDEFINED，从性能和资源占用的角度都会快很多。

```
enum class ImageUsageFlagBits : VkImageUsageFlags
  {
    eTransferSrc                     = VK_IMAGE_USAGE_TRANSFER_SRC_BIT,
    eTransferDst                     = VK_IMAGE_USAGE_TRANSFER_DST_BIT,
    eSampled                         = VK_IMAGE_USAGE_SAMPLED_BIT,
    eStorage                         = VK_IMAGE_USAGE_STORAGE_BIT,
    eColorAttachment                 = VK_IMAGE_USAGE_COLOR_ATTACHMENT_BIT,
    eDepthStencilAttachment          = VK_IMAGE_USAGE_DEPTH_STENCIL_ATTACHMENT_BIT,
    eTransientAttachment             = VK_IMAGE_USAGE_TRANSIENT_ATTACHMENT_BIT,
    eInputAttachment                 = VK_IMAGE_USAGE_INPUT_ATTACHMENT_BIT,
    eVideoDecodeDstKHR               = VK_IMAGE_USAGE_VIDEO_DECODE_DST_BIT_KHR,
    eVideoDecodeSrcKHR               = VK_IMAGE_USAGE_VIDEO_DECODE_SRC_BIT_KHR,
    eVideoDecodeDpbKHR               = VK_IMAGE_USAGE_VIDEO_DECODE_DPB_BIT_KHR,
    eShadingRateImageNV              = VK_IMAGE_USAGE_SHADING_RATE_IMAGE_BIT_NV,
    eFragmentDensityMapEXT           = VK_IMAGE_USAGE_FRAGMENT_DENSITY_MAP_BIT_EXT,
    eFragmentShadingRateAttachmentKHR = VK_IMAGE_USAGE_FRAGMENT_SHADING_RATE_ATTACHMENT_BIT_KHR,
#if defined( VK_ENABLE_BETA_EXTENSIONS )
    eVideoEncodeDstKHR = VK_IMAGE_USAGE_VIDEO_ENCODE_DST_BIT_KHR,
    eVideoEncodeSrcKHR = VK_IMAGE_USAGE_VIDEO_ENCODE_SRC_BIT_KHR,
    eVideoEncodeDpbKHR = VK_IMAGE_USAGE_VIDEO_ENCODE_DPB_BIT_KHR,
#endif /*VK_ENABLE_BETA_EXTENSIONS*/
    eAttachmentFeedbackLoopEXT = VK_IMAGE_USAGE_ATTACHMENT_FEEDBACK_LOOP_BIT_EXT,
    eInvocationMaskHUAWEI      = VK_IMAGE_USAGE_INVOCATION_MASK_BIT_HUAWEI,
    eSampleWeightQCOM          = VK_IMAGE_USAGE_SAMPLE_WEIGHT_BIT_QCOM,
    eSampleBlockMatchQCOM      = VK_IMAGE_USAGE_SAMPLE_BLOCK_MATCH_BIT_QCOM
  };
```

来个例子

比如一张Image ，写完成后转为可读，一般都是通过同步来操作，正如官方介绍的

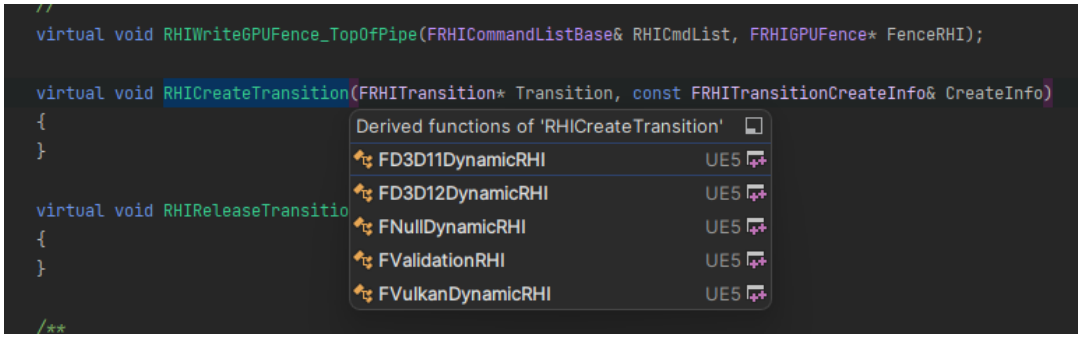
One of the most common ways to perform layout transitions is using an *image memory barrier*. A pipeline barrier like that is generally used to synchronize access to resources, like ensuring that a write to a buffer completes before reading from it, but it can also be used to transition image layouts and transfer queue family ownership when `VK_SHARING_MODE_EXCLUSIVE` is used. There is an equivalent *buffer memory barrier* to do this for buffers.

如下API：

```
command_buffer->pipelineBarrier(
    source_stage,
    destination_stage,
    vk::DependencyFlags(),
    0, nullptr,
    0, nullptr,
    1, &barrier
  );
```

其实在Unreal 中也有相关的代码：

```
RHICmdList.Transition({
        FRHITransitionInfo(ParticleSimulationResources->RenderAttributesTexture.TextureRHI, ERHIAccess::Unknown, ERHIAccess::RTV),
        FRHITransitionInfo(ParticleSimulationResources->SimulationAttributesTexture.TextureRHI, ERHIAccess::Unknown, ERHIAccess::RTV)
      });
```



居然Dx11也有这个功能。但看着只是改了bUAVBarrier，大胆猜测只是UE 内部的一些statu