4.3 初始化Direct3D

我们对Direct3D进行初始化的过程可以分为以下几个步骤：

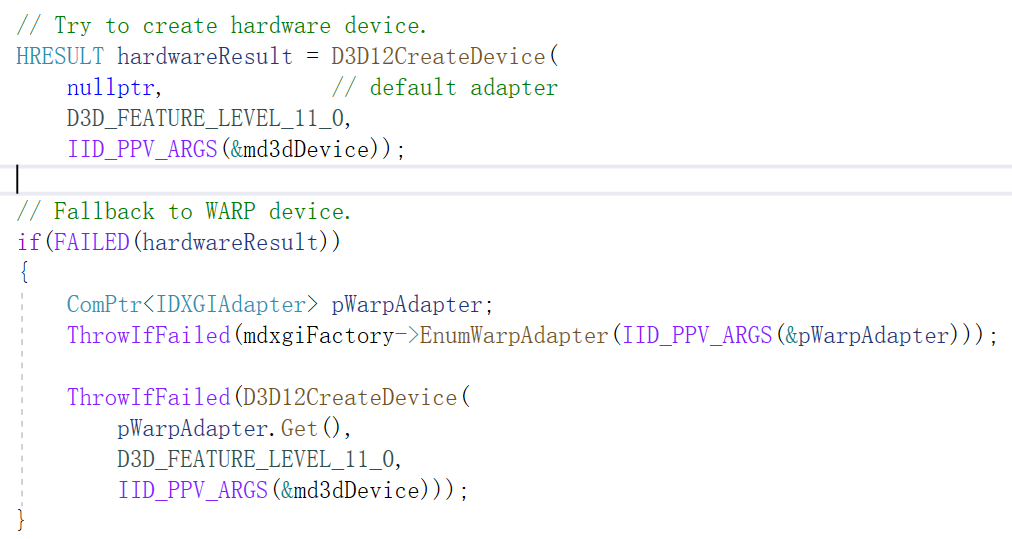
1. 创建设备
2. 创建围栏并获取描述符的大小
3. 检测对4X MSAA质量级别的支持
4. 创建命令队列和名列列表
5. 描述并创建交换链
6. 创建描述符堆
7. 创建渲染目标视图
8. 创建深度/模板缓冲区及其视图
9. 设置视口和裁剪矩形
10. **创建设备**

Direct3D12设备代表着一个显示适配器。一般来说，显示适配器是一种3D图形硬件，如显卡。但是，系统也能有软件显示适配器来模拟3D图形硬件的功能(如WARP适配器)。

创建Direct3D 12设备的函数：

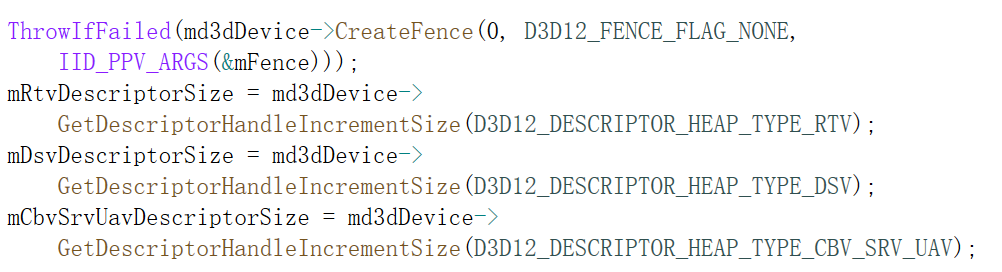
API：

D3D12CreateDevice

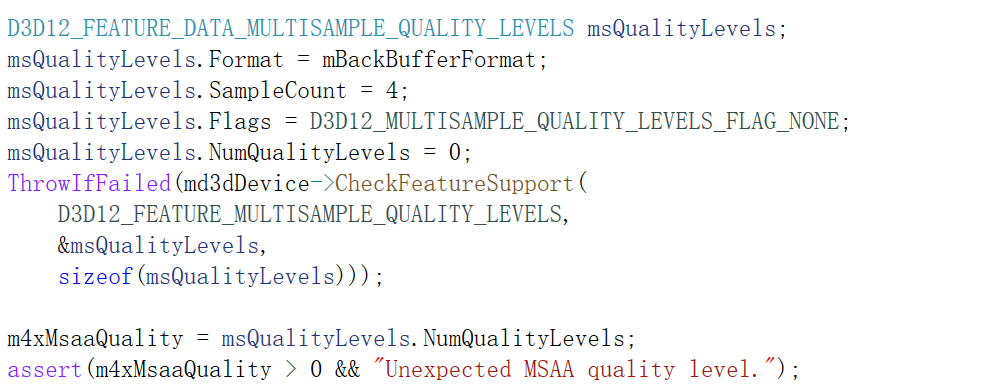


1. **创建围栏并获取描述符的大小**

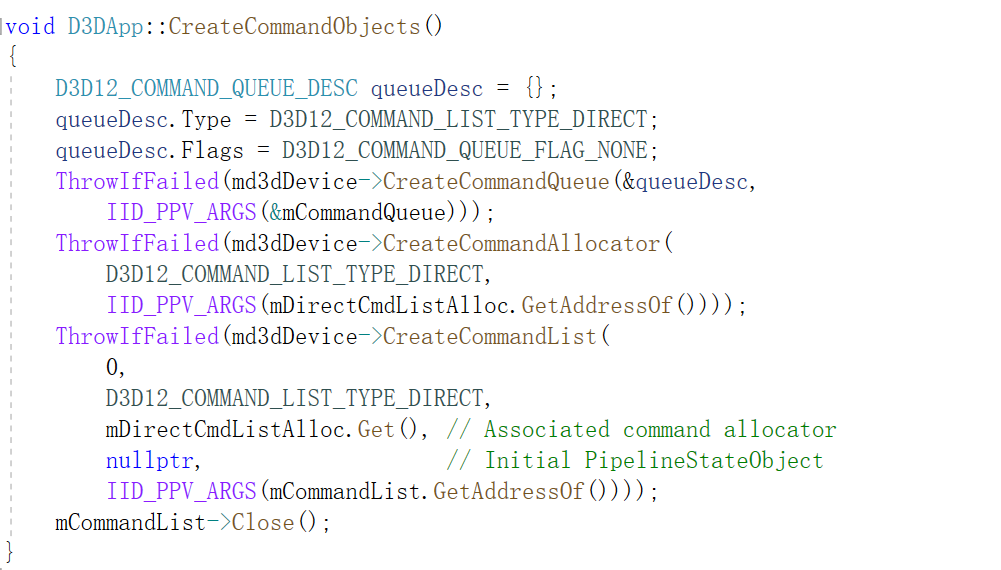
一旦创建好设备，便可以为CPU/GPU的同步而创建围栏了。另外，若用描述符进行工作，还需要了解它们的大小。



1. **检测对4X MSAA质量级别的支持**

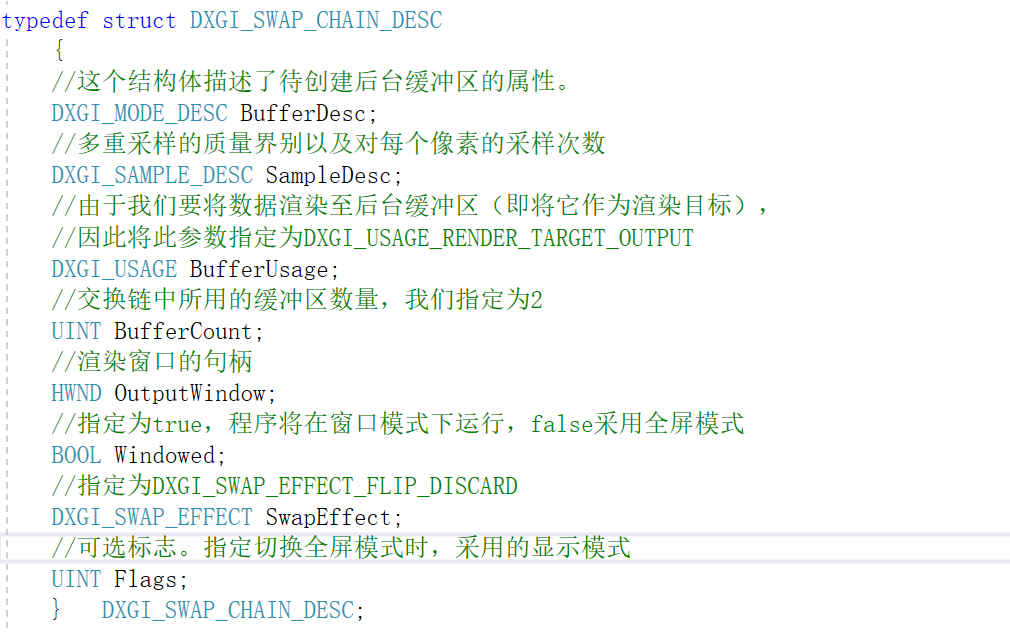


1. **创建命令队列和命令列表**

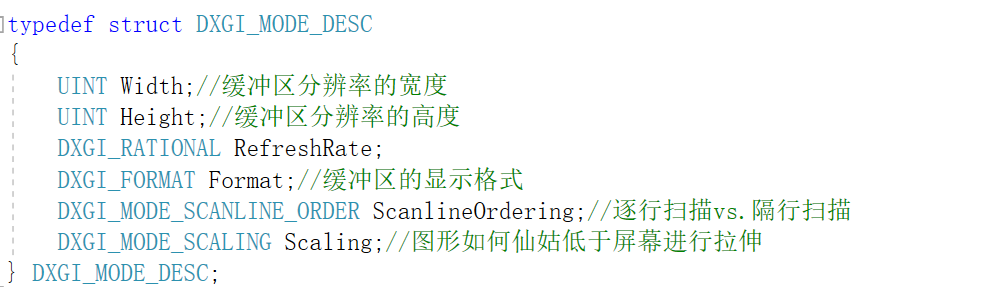


1. **描述并创建交换链**

创建交换链，首先要填写一份DXGI\_SWAP\_CHAIN\_DESC结构体，用它来描述欲创建交换链的特性。此结构体定义如下：



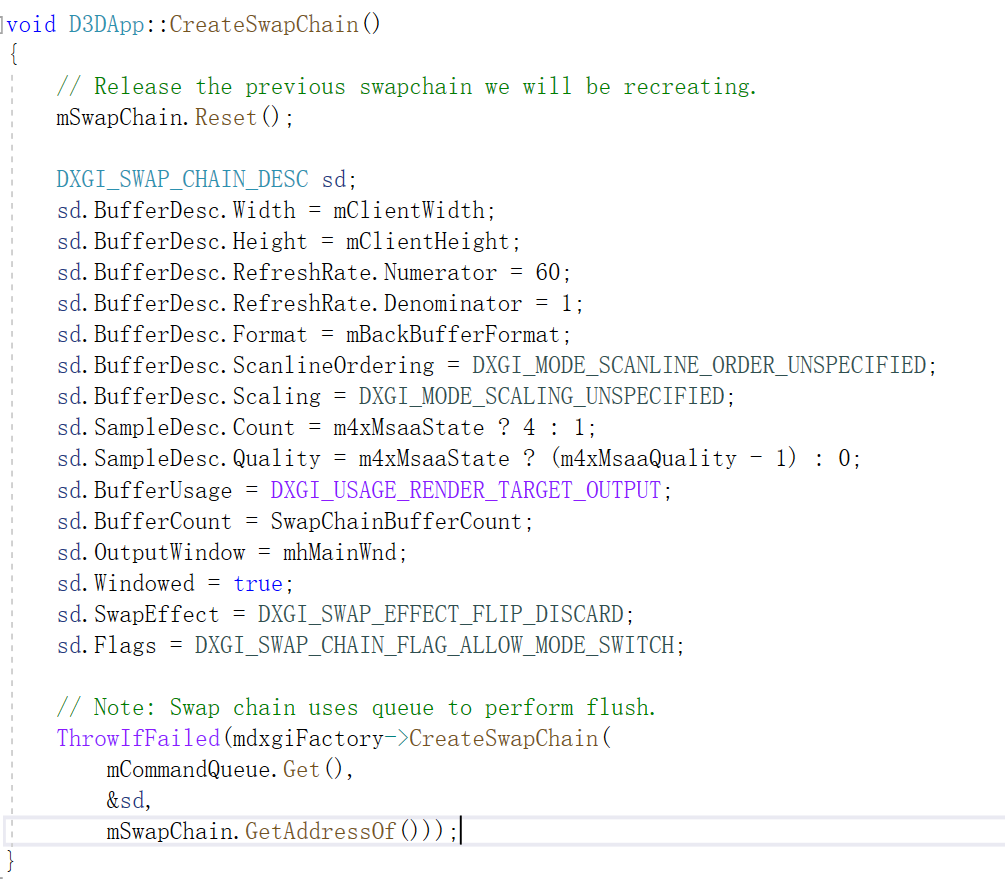
其中DXGI\_MODE\_DESC类型是另一种结构体，它的定义为：



创建交换链：



我们按照可以对交换链进行多次调用来设计的，即在创建新的交换链之前，先要销毁旧的交换链。

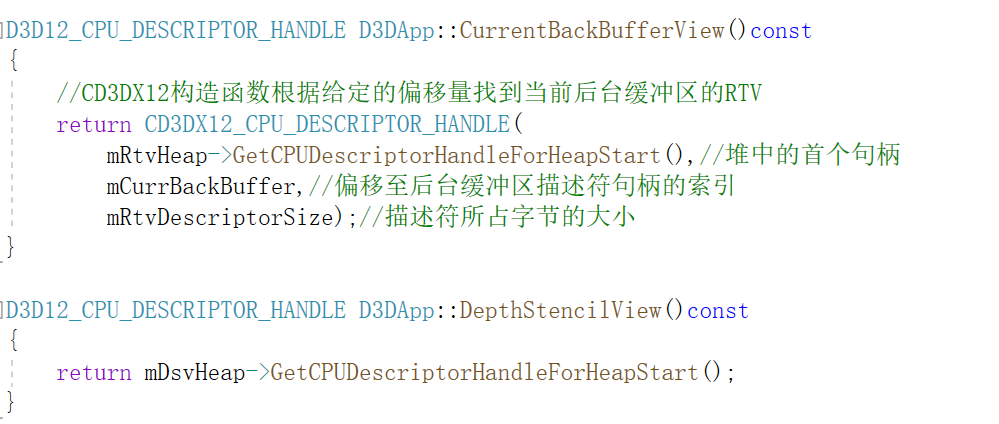


1. **创建描述符堆**

我们需要创建描述符堆来存储程序中要用到的描述符。我们需要创建两个描述符堆，其一用来存储SwapChainBufferCount个RTV，而那另一个描述符堆则用来存储那1个DSV。

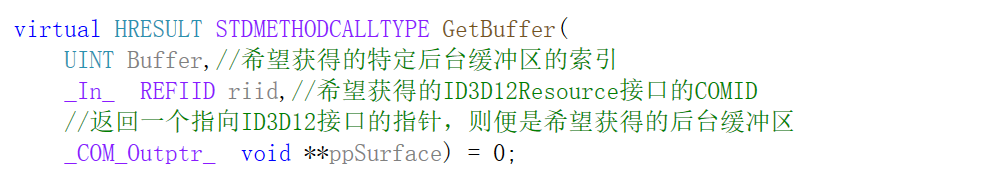


创建描述符堆之后，还要能访问其中所存的描述符。mCurrBackBuffer用来记录当前后台缓冲区的索引（记录那个缓冲区才是当前正在用于渲染数据的后台缓冲区）。

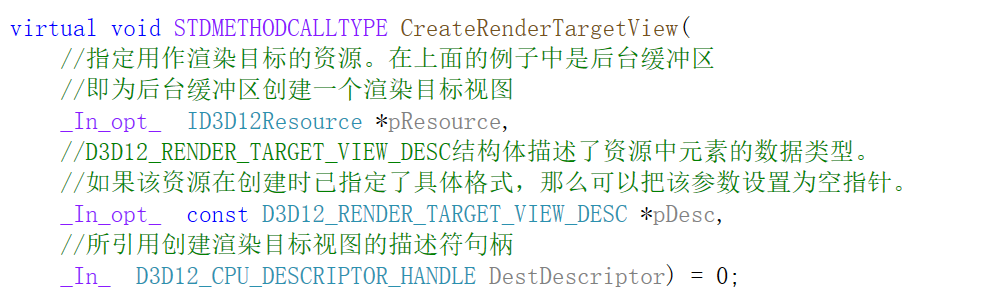


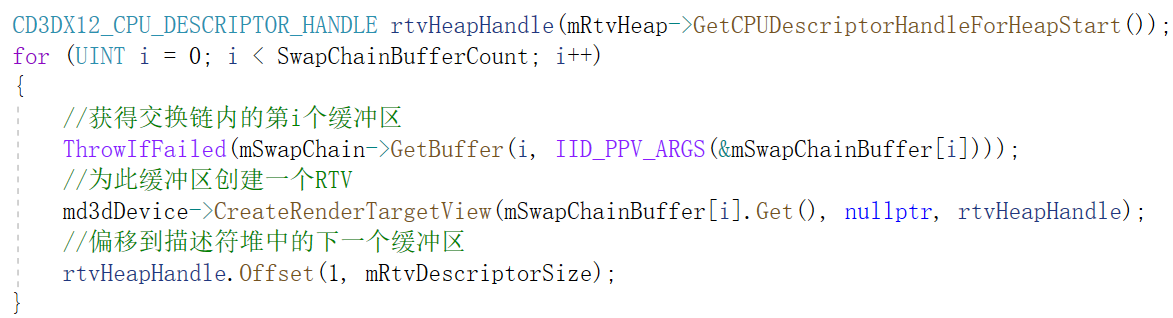
1. **创建渲染目标视图**

资源不能与渲染流水线中的阶段直接绑定，所以我们必须先为资源创建视图。并将其绑定到流水线阶段。例如，为了将后台缓冲区绑定到流水线的输出合并阶段，便需要为该后台缓冲区创建一个渲染目标视图，而第一步就是要获得存于交换链中的缓冲区资源。



接下来，使用ID3D12Device::CreateRenderTargetView方法来为获取后台缓冲区创建渲染目标视图。

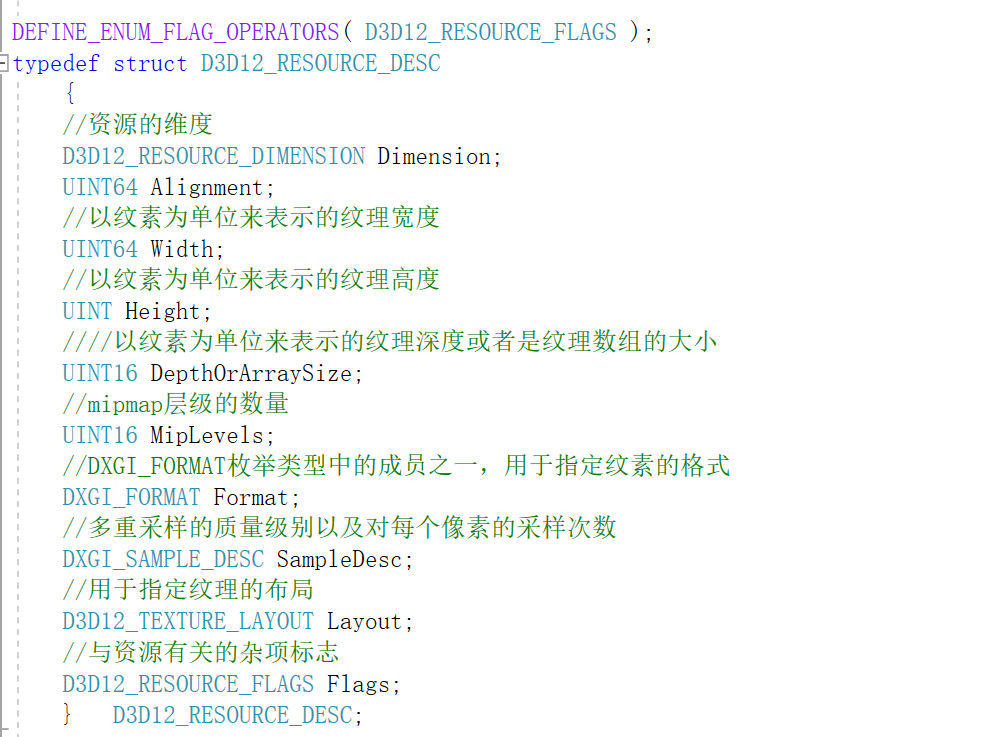




1. **创建深度/模板缓冲区及其视图**

深度缓冲区其实就是一种2D纹理，它存储着离观察者最近的可视对象的深度信息。纹理是一种GPU资源，因此我们要通过填写D3D12\_RESOURCE\_DESC结构体来描述纹理资源。再用ID3D12Device::CreateCommittedResource方法来创建它。

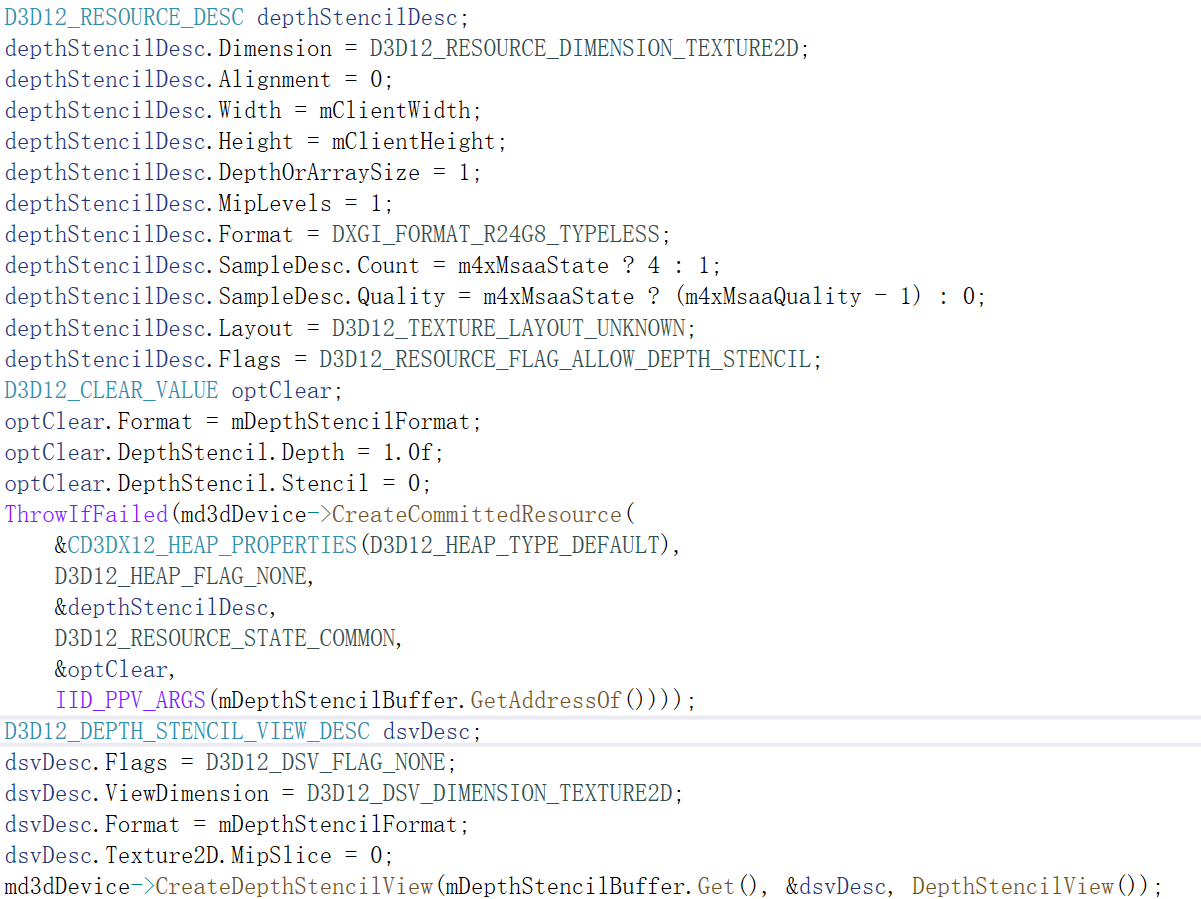
D3D12\_RESOURCE\_DESC结构体定义如下：



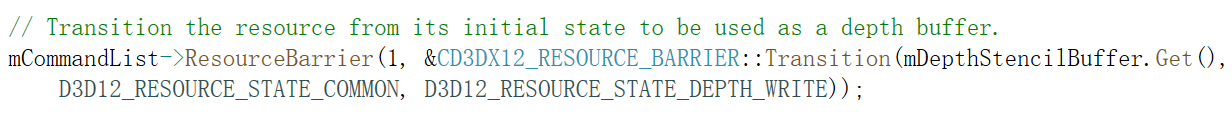
GPU资源都存在于堆中，其本质是具有特定属性的GPU显存块。ID3D12Device::CreateCommittedResource方法将根据我们所提供的属性创建一个资源与一个堆，并把该资源提交到这个堆中。



另外，在使用深度/模板缓冲区之前，一定要创建相关的深度/模板视图，并将它绑定到渲染流水线上。



将资源从初始状态转换为深度缓冲区



**9 设置视口**

我们通常会将3D场景绘制到与整个屏幕（在全屏模式下）或整个窗口工作区大小相当的后台缓冲区中。但是，有时只是希望把3D场景绘制到后台缓冲区的某个矩形子区域当中。

我们把后台缓冲区中的这种矩形子区域叫做视口。



结构体中的前4个数据成员定义了视口矩形相对于后台缓冲区的绘制范围。在Direct3D中，存储在深度缓冲区中的数据范围都在0-1内，MinDepth和MaxDepth这两个成员将[0,1]转换至[MinDepth,MaxDepth]。

只要填写好D3D12\_VIEWPORT结构体，便可以用ID3D12GraphicsCommandList::

RSSetViewports方法来设置视口。

事实上，还可以用视口技术来实现双人游戏的分屏模式。

**10，设置裁剪矩形**

我们可以相对于后台缓冲区定义一个裁剪矩形，在此矩形外的像素都将被剔除（即这些图像部分将不会被光栅化至后台缓冲区）。这个方法能用于优化程序的性能。例如，假设已知有一个矩形的UI元素覆于屏幕中某块区域的最上层，那么我们就无需对3D空间中哪些被他遮挡的像素进行处理类。

裁剪矩形有类型为D3D12\_RECT结构体定义而成：

在Direct3D中，要用ID3D12GraphicsCommandList::RSSetCissorRects方法来设置裁剪矩形。

下面示例将创建并设置一个覆盖后台缓冲区左上角1/4区域的剪裁矩形

mScissorRect = { 0, 0, mClientWidth / 2, mClientHeight / 2};

mCommandList->RSSetScissorRects(1, &mScissorRect)

不能为同一个渲染目标指定多个裁剪矩形。