黑客派

DX12 之 miniEngine 代码自解

作者：xu365082218

原文链接：https://hacpai.com/article/1510829924240

来源网站：黑客派

```
<p>好像有个叫 Vulkan 的图形 API 试图替代 Opengl，不过要跟上时代，还是得把 WIN10 装上，
试看在 DX12 写个下个时代的 3D 设计工具，当然只是个小玩意，不过据说很多软件能走到今天，也
因为积淀，如果不早点开始积淀，就永远会落后那么多<br> 我想给这个玩意起个名字，叫啥呢，就叫
W4D 好了，首先这个软件定位不是游戏引擎，而且是超轻量的，W：windows 4：x y z t D： Data
指代在 windows 下处理 3 维与时间数据的软件<br> 想要有什么呢？<br> 1：首先需要能应对基本
 3D 建模，绑骨，展 UV 贴图，动作<br> 2：要有语言能应对复杂的各种情况 类似 maxscript 脚本
言，但是这里可能级别要更强<br> 3：最好有时间轴，觉得这个概念与 2D 里 flash 接近，跟 unity
 animation 差不多，用不着搞成 timeline 那么复杂。<br> 代码使用 C++ DX，确定要一切考虑效
，即使函数入口都不要检查，只在最外层进行检查，出错是程序员的事，代码不是用来为程序员负责
，希望这个玩意能像刀片一样 欻欻的就搞定哈哈</p>
<script async src="https://pagead2.googlesyndication.com/pagead/js/adsbygoogle.js"></scr
pt>
<!-- 黑客派PC帖子内嵌-展示 -->
<ins class="adsbygoogle" style="display:block" data-ad-client="ca-pub-5357405790190342"
data-ad-slot="8316640078" data-ad-format="auto" data-full-width-responsive="true"></in
>
<script>
    (adsbygoogle = window.adsbygoogle || []).push({});
</script>
<p>环境 WIN10,VS2017.这二者搞定后，微软官方例子可以直接打开编译运行的，miniEngine 可
要重定向项目-&gt; 选择相应的 SDK 版本（在解决方案右键会出来选项）</p>
<p>dx12 基础<br> <a href="https://link.hacpai.com/forward?goto=http%3A%2F%2Fwww.ras
ertek.com%2Ftutdx12.html" target="_blank" rel="nofollow ugc">http://www.rastertek.com/tu
dx12.html</a><br> 官网的<br> <a href="https://link.hacpai.com/forward?goto=https%3A%
F%2Fmsdn.microsoft.com%2Fen-us%2Flibrary%2Fwindows%2Fdesktop%2Fdn899121%28v%
Dvs.85%29.aspx" target="_blank" rel="nofollow ugc">https://msdn.microsoft.com/en-us/libra
y/windows/desktop/dn899121(v=vs.85).aspx</a></p>
<h2 id="Working-samples">Working samples</h2>
<p>例子<br> Working samples (in the form of Visual Studio 2015 projects) can be download
d from <a href="https://link.hacpai.com/forward?goto=https%3A%2F%2Fgithub.com%2FMic
osoft%2FDirectX-Graphics-Samples" target="_blank" rel="nofollow ugc">GitHub/Microsoft/D
rectX-Graphics-Samples</a>.</p>
<p>DX12 遍历显卡<br> Intel(R) HD Graphics 4600<br> NVIDIA GeForce GTX 960M<br> Mic
osoft Basic Render Driver</p>
<p>把特性设置为 D3D_FEATURE_LEVEL_11_0</p>
<p>HRESULT hr = CreateDXGIFactory(__uuidof(IDXGIFactory4), (void**)&amp;factory);<br> in
 i = 0;<br> IDXGIAdapter * pAdapter;<br> std::vector &lt;IDXGIAdapter*&gt; vAdapters;<br>
while (factory-&gt;EnumAdapters(i, &amp;pAdapter) != DXGI_ERROR_NOT_FOUND)<br> {<b
> //hr = D3D12CreateDevice(pAdapter, feature_level, __uuidof(ID3D12Device), (void**)&amp;
pDevice);<br> //if (SUCCEEDED(hr))<br> // break;<br> pAdapter-&gt;GetDesc(&amp;adapte
Desc);<br> wcstombs_s(&amp;stringLength, (char*)videoCardDes, 128, adapterDesc.Descripti
n, 128);<br> vAdapters.push_back(pAdapter);<br> ++i;<br> }</p>
<p><a href="https://link.hacpai.com/forward?goto=http%3A%2F%2Fwww.rastertek.com%2F
 target="_blank" rel="nofollow ugc">http://www.rastertek.com/</a><br> 这个网站上 3 篇 DX
2 的已经照着写了一遍，感觉概念多，复杂，勉强能跑个背景色出来就废了这么大功夫，可惜的是作
似乎没有继续写了<br> 还是看 dx12 在 Git 上的例子好了。</p>
<p>哎呀，感觉好难啊。<br> 打开微软 ModelViewer_VS15.sln 的例子，运行效果还是挺好的，就
连入口都摸了半天<br> <img src="https://static.hacpai.com/images/img-loading.svg" alt="场
示例" data-src="http://www.idevgame.com/images/article/minigraph1.png"></p>
<p>入口是一个宏<br> CREATE_APPLICATION( ModelViewer )</p>
<p>映射到 GameCore::RunApplication( app_class(), L#app_class );<br> namespace GameCore
里定义了这个全局函数 RunApplication<br> 这个函数初始化了引擎的各个部分，注册了窗口类，
建了消息循环，循环结束后，进行各个部分的资源清理，基本所有引擎框架都有类似这样的一个结构
。<br> 不过这个函数的入口是 c 语言的 int wmain(int argc, wchar_t** argv)，估计是要通过宏适
```

多平台吧。

在 GameCore::RunApplication 里看下有哪些模块都在这里初始化的
 在创建完窗口后调
了
 InitializeApplication(app);
 定义为
 void InitializeApplication(IGameApp&amp
game)
 {
 Graphics::Initialize();
 SystemTime::Initialize();
 GameInput::Initiali
e();
 EngineTuning::Initialize();
<pre><code class="highlight-chroma">game.Startup();
</code></pre>
}
 这个 game 参数，就是 ModelViewer 的一个对象
这部分基本和
 <a href="https://link.hacpai.com/forward?goto=http%3A%2F%2Fwww.
astertek.com" target="_blank" rel="nofollow ugc">www.rastertek.com
 描述的差不
，图形初始化，输入初始化，游戏初始化，系统时间初始化，引擎协调初始化
 Graphics::Initiali
e();在这里，
 创建了 dx12 设备
 (遍历了一次取了显存最大的显卡，还过滤掉了软件模拟卡

SIZE_T MaxSize = 0;
for (uint32_t Idx = 0; DXGI_ERROR_NOT_FOUND != dxgiFactory->EnumAdapters1(Idx,
&pAdapter); ++Idx)
 {
 DXGI_ADAPTER_DESC1 desc;
 pAdapter->GetDes
1(&desc);
 if (desc.Flags & DXGI_ADAPTER_FLAG_SOFTWARE)
 continue;</
>
<pre><code class="highlight-chroma">if (desc.DedicatedVideoMemory > MaxSize &amp
& SUCCEEDED(D3D12CreateDevice(pAdapter.Get(), D3D_FEATURE_LEVEL_11_0, MY_IID_
PV_ARGS(&pDevice))))
{
    pAdapter->GetDesc1(&desc);
    Utility::Printf(L"D3D12-capable hardware found: %s (%u MB)\n", desc.Description, desc.Ded
catedVideoMemory >> 20);
    MaxSize = desc.DedicatedVideoMemory;
}
</code></pre>
}
if (MaxSize > 0)
 g_Device = pDevice.Detach();
然后检查了设备是否支持 R11G11B10 和 R16G16B16A16 2 种 UAVLoadSupport 上传格式，
部分仅仅设置了几个变量标识是否有此特性（g_bTypedUAVLoadSupport_R11G11B10_FLOAT，g_
TypedUAVLoadSupport_R16G16B16A16_FLOAT）。
D3D12_FEATURE_DATA_D3D12_OPTIONS FeatureData = {};
 if (SUCCEEDED(g_Device
>CheckFeatureSupport(D3D12_FEATURE_D3D12_OPTIONS, &FeatureData, sizeof(Feat
reData))))
 {
 if (FeatureData.TypedUAVLoadAdditionalFormats)
 {
 D3D12_F
ATURE_DATA_FORMAT_SUPPORT Support =
 {
 DXGI_FORMAT_R11G11B10_FLOAT,
3D12_FORMAT_SUPPORT1_NONE, D3D12_FORMAT_SUPPORT2_NONE
 };

(adsbygoogle = window.adsbygoogle || []).push({});
</script>
<pre><code class="highlight-chroma">   if (SUCCEEDED(g_Device->CheckFeatureSuppor
(D3D12_FEATURE_FORMAT_SUPPORT, &Support, sizeof(Support))) &&
        (Support.Support2 & D3D12_FORMAT_SUPPORT2_UAV_TYPED_LOAD) != 0)
    {
        g_bTypedUAVLoadSupport_R11G11B10_FLOAT = true;
    }

}
</code></pre>

}
接着创建了各种 commandqueue，
 g_CommandManager.Create(g_Device);
 g_C
mmandManager 是 CommandListManager 类型的
 跟进去看一下
 void CommandList
anager::Create(ID3D12Device* pDevice)
 {
 ASSERT(pDevice != nullptr);
<pre><code class="highlight-chroma">m_Device = pDevice;

m_GraphicsQueue.Create(pDevice);
m_ComputeQueue.Create(pDevice);
m_CopyQueue.Create(pDevice);
</code></pre>

}
m_GraphicsQueue m_ComputeQueue m_CopyQueue 都是 CommandQueue，这里的实现
一样
void CommandQueue::Create(ID3D12Device* pDevice)
 {
 ASSERT(pDevice != nul
ptr);
 ASSERT(!IsReady());
 ASSERT(m_AllocatorPool.Size() == 0);
<pre><code class="highlight-chroma">D3D12_COMMAND_QUEUE_DESC QueueDesc = {};
QueueDesc.Type = m_Type;
QueueDesc.NodeMask = 1;
pDevice->CreateCommandQueue(&QueueDesc, MY_IID_PPV_ARGS(&m_Comm
ndQueue));
m_CommandQueue->SetName(L"CommandListManager::m_CommandQueue");

ASSERT_SUCCEEDED(pDevice-not found render function for node [type=NodeHTMLEntity, To
ens=>]not found render function for node [type=NodeHTMLEntity, Tokens=>]Create
ence(0, D3D12_FENCE_FLAG_NONE, MY_IID_PPV_ARGS(not found render function for node [t
pe=NodeHTMLEntity, Tokens=&]not found render function for node [type=NodeHTMLEntity,
Tokens=&]m_pFence)));
m_pFence-not found render function for node [type=NodeHTMLEntity, Tokens=>]
ot found render function for node [type=NodeHTMLEntity, Tokens=>]SetName(L"CommandL
stManager::m_pFence");
m_pFence-not found render function for node [type=NodeHTMLEntity, Tokens=>]
ot found render function for node [type=NodeHTMLEntity, Tokens=>]Signal((uint64_t)m_Typ
 not found render function for node [type=NodeHTMLEntity, Tokens=<]not found render fu
ction for node [type=NodeHTMLEntity, Tokens=<]not found render function for node [type
NodeHTMLEntity, Tokens=<]not found render function for node [type=NodeHTMLEntity, Tok
ns=<] 56);

```
m_FenceEventHandle = CreateEvent(nullptr, false, false, nullptr);
ASSERT(m_FenceEventHandle != INVALID_HANDLE_VALUE);

m_AllocatorPool.Create(pDevice);

ASSERT(IsReady());
</code></pre>

<script async src="https://pagead2.googlesyndication.com/pagead/js/adsbygoogle.js"></scr
pt>

<!-- 黑客派PC帖子内嵌-展示 -->

<ins class="adsbygoogle" style="display:block" data-ad-client="ca-pub-5357405790190342"
data-ad-slot="8316640078" data-ad-format="auto" data-full-width-responsive="true"></in
>
<script>
    (adsbygoogle = window.adsbygoogle || []).push({});
</script>
<p>}</p>
<p>这个类应该算对 dx12 commandqueue 的封装吧，核心就是使用 pDevice-&gt;CreateComma
dQueue(&amp;QueueDesc, MY_IID_PPV_ARGS(&amp;m_CommandQueue));<br> 然后创建了 f
nce,fence_event<br> m_AllocatorPool.Create(pDevice);只是把类的成员变量设置为 pDevice，
个 m_AllocatorPool 也是对 commandAllocator 的一个池的封装 他成员里有 std::vector&lt;ID3D1
CommandAllocator*&gt; m_AllocatorPool;<br> 这部分完了，返回堆栈上面的 Graphics::Initialize
);</p>
<p>在 g_CommandManager.Create(g_Device);之后<br> 有</p>
<p>DXGI_SWAP_CHAIN_DESC1 swapChainDesc = {};<br> swapChainDesc.Width = g_Displa
Width;<br> swapChainDesc.Height = g_DisplayHeight;<br> swapChainDesc.Format = SwapC
ainFormat;<br> swapChainDesc.Scaling = DXGI_SCALING_NONE;<br> swapChainDesc.Sampl
Desc.Quality = 0;<br> swapChainDesc.SampleDesc.Count = 1;<br> swapChainDesc.BufferUs
ge = DXGI_USAGE_RENDER_TARGET_OUTPUT;<br> swapChainDesc.BufferCount = SWAP_CH
IN_BUFFER_COUNT;<br> swapChainDesc.Flags = DXGI_SWAP_CHAIN_FLAG_ALLOW_MODE_
WITCH;<br> swapChainDesc.SwapEffect = DXGI_SWAP_EFFECT_FLIP_SEQUENTIAL;</p>
<p>#if WINAPI_FAMILY_PARTITION(WINAPI_PARTITION_DESKTOP) // Win32<br> ASSERT_S
CCEEDED(dxgiFactory-&gt;CreateSwapChainForHwnd(g_CommandManager.GetCommandQu
ue(), GameCore::g_hWnd, &amp;swapChainDesc, nullptr, nullptr, &amp;s_SwapChain1));</p>
<p>创建了一个交换链的描述，然后通过这个描述(swapChainDesc)创建了一个交换链，g_Comman
Manager.GetCommandQueue()返回的是他成员里，m_GraphicsQueue 创建的 CommandQueue,
交换链接口的指针放到 s_SwapChain1 里<br> 下面有部分代码不明白意思，跳过去</p>
<p>#if CONDITIONALLY_ENABLE_HDR_OUTPUT &amp;&amp; defined(NTDDI_WIN10_RS2)
amp;&amp; (NTDDI_VERSION &gt;= NTDDI_WIN10_RS2)<br> {<br> IDXGISwapChain4* swa
Chain = (IDXGISwapChain4*)s_SwapChain1;<br> ComPtr output;<br> ComPtr output6;<br>
XGI_OUTPUT_DESC1 outputDesc;<br> UINT colorSpaceSupport;</p>
<script async src="https://pagead2.googlesyndication.com/pagead/js/adsbygoogle.js"></scr
pt>
<!-- 黑客派PC帖子内嵌-展示 -->
<ins class="adsbygoogle" style="display:block" data-ad-client="ca-pub-5357405790190342"
data-ad-slot="8316640078" data-ad-format="auto" data-full-width-responsive="true"></in
>
<script>
    (adsbygoogle = window.adsbygoogle || []).push({});
</script>
```

<pre><code class="highlight-chroma">// Query support for ST.2084 on the display and set t
e color space accordingly
if (SUCCEEDED(swapChain->GetContainingOutput(&output)) &&
    SUCCEEDED(output.As(&output6)) &&
    SUCCEEDED(output6->GetDesc1(&outputDesc)) &&
    outputDesc.ColorSpace == DXGI_COLOR_SPACE_RGB_FULL_G2084_NONE_P2020 &&
mp;
    SUCCEEDED(swapChain->CheckColorSpaceSupport(DXGI_COLOR_SPACE_RGB_FULL_G2
84_NONE_P2020, &colorSpaceSupport)) &&
    (colorSpaceSupport & DXGI_SWAP_CHAIN_COLOR_SPACE_SUPPORT_FLAG_PRESENT)
&&
    SUCCEEDED(swapChain->SetColorSpace1(DXGI_COLOR_SPACE_RGB_FULL_G2084_NON
_P2020)))
{
    g_bEnableHDROutput = true;
}
</code></pre>
}
 #endif
 看英文解释是是否支持 ST.2048 并设置交换链的颜色空间的。
接着看
 for (uint32_t i = 0; i < SWAP_CHAIN_BUFFER_COUNT; ++i)
 {
 Co
Ptr DisplayPlane;
 ASSERT_SUCCEEDED(s_SwapChain1->GetBuffer(i, MY_IID_PPV_ARGS
&DisplayPlane)));
 g_DisplayPlane[i].CreateFromSwapChain(L"Primary SwapChain Buf
er", DisplayPlane.Detach());
 }
这部分是创建交换链使用的资源以及相应的 RenderTargetView，一个交换链上创建数个 RTV
话说这个包装还是有点难理解的。
 交换链上 GetBuffer 类似按照指定的序号（在这里就是 i）
把 ID3D12Resource 指针返回，要使用就创建一个 ID3D12Resource 描述的 RTV 对象
 这个 R
V 对象是通过
 g_DisplayPlane[i].CreateFromSwapChain(L"Primary SwapChain Buffer", Disp
ayPlane.Detach())创建的
 g_DisplayPlane 是一个 ColorBuffer 数组
 跟进去看一下
void ColorBuffer::CreateFromSwapChain(const std::wstring& Name, ID3D12Resourc
* BaseResource)
 {
 AssociateWithResource(Graphics::g_Device, Name, BaseResource
 D3D12_RESOURCE_STATE_PRESENT);

<pre><code class="highlight-chroma">//m_UAVHandle[0] = Graphics::AllocateDescriptor(D
D12_DESCRIPTOR_HEAP_TYPE_CBV_SRV_UAV);
//Graphics::g_Device->CreateUnorderedAccessView(m_pResource.Get(), nullptr, nullptr, m
UAVHandle[0]);

m_RTVHandle = Graphics::AllocateDescriptor(D3D12_DESCRIPTOR_HEAP_TYPE_RTV);

Graphics::g_Device-not found render function for node [type=NodeHTMLEntity, Tokens=>]
ot found render function for node [type=NodeHTMLEntity, Tokens=>]CreateRenderTargetVi
w(m_pResource.Get(), nullptr, m_RTVHandle);

</code></pre>

}
再跟进
 AssociateWithResource(Graphics::g_Device, Name, BaseResource, D3D12_RE
OURCE_STATE_PRESENT);

ColorBuffer 继承 PixelBuffer，跳转到这里
void PixelBuffer::AssociateWithResource(ID3D12Device* Device, const std::wstring& Name, ID3D12Resource* Resource, D3D12_RESOURCE_STATES CurrentState)
 {
 (Device); // Unused until we support multiple adapters
<pre><code class="highlight-chroma">ASSERT(Resource != nullptr);
D3D12_RESOURCE_DESC ResourceDesc = Resource->GetDesc();

m_pResource.Attach(Resource);

m_UsageState = CurrentState;

m_Width = (uint32_t)ResourceDesc.Width; // We don't care about large virtual textures yet

m_Height = ResourceDesc.Height;

m_ArraySize = ResourceDesc.DepthOrArraySize;

m_Format = ResourceDesc.Format;

</code></pre>

#ifndef RELEASE
 m_pResource->SetName(Name.c_str());
 #else
 (Name)
 #endif
 }
这个函数只是调用 ID312DResources 指针获取了资源描述 GetDesc()，然后把这些描述的属性在了自己的成员里
 跳出 AssociateWithResource 函数返回
 void ColorBuffer::CreateFroSwapChain(const std::wstring& Name, ID3D12Resource* BaseResource)
 {
 AssociateWithResource(Graphics::g_Device, Name, BaseResource, D3D12_RESOURCE_STATE_PRESNT);//跳出堆栈
<script async src="https://pagead2.googlesyndication.com/pagead/js/adsbygoogle.js"></script>
<!-- 黑客派PC帖子内嵌-展示 -->
<ins class="adsbygoogle" style="display:block" data-ad-client="ca-pub-5357405790190342" data-ad-slot="8316640078" data-ad-format="auto" data-full-width-responsive="true">
<script>
    (adsbygoogle = window.adsbygoogle || []).push({});
</script>
<pre><code class="highlight-chroma">//m_UAVHandle[0] = Graphics::AllocateDescriptor(D12_DESCRIPTOR_HEAP_TYPE_CBV_SRV_UAV);
//Graphics::g_Device->CreateUnorderedAccessView(m_pResource.Get(), nullptr, nullptr, mUAVHandle[0]);
 //到这里
m_RTVHandle = Graphics::AllocateDescriptor(D3D12_DESCRIPTOR_HEAP_TYPE_RTV);
Graphics::g_Device->CreateRenderTargetView(m_pResource.Get(), nullptr, m_RTVHandle);
</code></pre>
}
m_RTVHandle 大概是一个 RTV 对象描述 的句柄。
 跟进 Graphics::AllocateDescriptor(D12_DESCRIPTOR_HEAP_TYPE_RTV);内部
 inline D3D12_CPU_DESCRIPTOR_HANDLE AllocteDescriptor(D3D12_DESCRIPTOR_HEAP_TYPE Type, UINT Count = 1)
 {
 return g_DscriptorAllocator[Type].Allocate(Count);
 }
 实际上最后是用 g_DescriptorAllocator[Typ].Allocate(Count)返回了一个句柄，看下 g_DescriptorAllocator 定义
 DescriptorAllocator g_escriptorAllocator[D3D12_DESCRIPTOR_HEAP_TYPE_NUM_TYPES] =
 {
 D3D12_DESCIPTOR_HEAP_TYPE_CBV_SRV_UAV,
 D3D12_DESCRIPTOR_HEAP_TYPE_SAMPLER,
 D312_DESCRIPTOR_HEAP_TYPE_RTV,
 D3D12_DESCRIPTOR_HEAP_TYPE_DSV,
 };
 的到这是一个 DescriptorAllocator 类型的数组，这个 g_DescriptorAllocator 数组有 4 个成员，每成员都是一个
 DescriptorAllocator 类型对象，这个类型对堆申请以及句柄申请做了一个池的

装
跟到 Allocate 里看一下
 D3D12_CPU_DESCRIPTOR_HANDLE DescriptorAllocator::Allocte(uint32_t Count)
 {
 if (m_CurrentHeap == nullptr || m_RemainingFreeHandles <Count)
 {
 m_CurrentHeap = RequestNewHeap(m_Type);
 m_CurrentHandle = mCurrentHeap->GetCPUDescriptorHandleForHeapStart();
 m_RemainingFreeHandles = m_NumDescriptorsPerHeap;

<pre><code class="highlight-chroma"> if (m_DescriptorSize == 0)
    m_DescriptorSize = Graphics::g_Device->GetDescriptorHandleIncrementSize(m_Type);
}

D3D12_CPU_DESCRIPTOR_HANDLE ret = m_CurrentHandle;

m_CurrentHandle.ptr += Count * m_DescriptorSize;

m_RemainingFreeHandles -= Count;

return ret;

</code></pre>

}
没有堆就创建一个堆
 跟到 m_CurrentHeap = RequestNewHeap(m_Type);这里
 参数 m_Type 就跟 g_DescriptorAllocator[Type]的 Type 一样就是 D3D12_DESCRIPTOR_HEAP_TYPE_RV
ID3D12DescriptorHeap* DescriptorAllocator::RequestNewHeap(D3D12_DESCRIPTOR_HEP_TYPE Type)
 {
 std::lock_guard LockGuard(sm_AllocationMutex);
<pre><code class="highlight-chroma">D3D12_DESCRIPTOR_HEAP_DESC Desc;
Desc.Type = Type;
Desc.NumDescriptors = sm_NumDescriptorsPerHeap;//256个描述
Desc.Flags = D3D12_DESCRIPTOR_HEAP_FLAG_NONE;
Desc.NodeMask = 1;

Microsoft::WRL::ComPtrnot found render function for node [type=NodeHTMLEntity, Tokens=]not found render function for node [type=NodeHTMLEntity, Tokens=<]ID3D12DescriptorHapnot found render function for node [type=NodeHTMLEntity, Tokens=>]not found render function for node [type=NodeHTMLEntity, Tokens=>] pHeap;

ASSERT_SUCCEEDED(Graphics::g_Device-not found render function for node [type=NodeHTLEntity, Tokens=>]not found render function for node [type=NodeHTMLEntity, Tokens=>]reateDescriptorHeap(not found render function for node [type=NodeHTMLEntity, Tokens=&]ot found render function for node [type=NodeHTMLEntity, Tokens=&]Desc, MY_IID_PPV_ARS(not found render function for node [type=NodeHTMLEntity, Tokens=&]not found renderunction for node [type=NodeHTMLEntity, Tokens=&]pHeap)));

sm_DescriptorHeapPool.emplace_back(pHeap);

return pHeap.Get();

</code></pre>

}
函数实现还带锁，那应该是提供给多线程使用的，这里也就是创建了一个堆描述对象，这个堆描对象的 NumDescriptors 为 256（sm_NumDescriptorsPerHeap = 256）
 最后还是通过
 g_Device->CreateDescriptorHeap(&Desc, MY_IID_PPV_ARGS(&pHeap);
 创了一个描述堆。
 返回了创建的 ID3D12DescriptorHeap 指针
 m_CurrentHandle = m_CurrentHeap->GetCPUDescriptorHandleForHeapStart();
 这个意思就是取得描述堆起始的句
 if (m_DescriptorSize == 0)
 m_DescriptorSize = Graphics::g_Device->GetDescriporHandleIncrementSize(m_Type);
 这个就是取得每 2 个相邻句柄的指针要偏移多少

<pre><code class="highlight-chroma">D3D12_CPU_DESCRIPTOR_HANDLE ret = m_Current
andle;
m_CurrentHandle.ptr += Count * m_DescriptorSize;
m_RemainingFreeHandles -= Count;
return ret;
</code></pre>
这个就是说 Count 参数，申请多少个句柄，用 ret 存储返回的句柄，把原句柄按照数量偏移到位置去。返回 ret（保存着偏移前的位置的句柄）
返回到
 m_RTVHandle = Graphics::AllocateDescriptor(D3D12_DESCRIPTOR_HEAP_TYE_RTV);
 接着
 Graphics::g_Device->CreateRenderTargetView(m_pResource.Get(), ullptr, m_RTVHandle);
 这句话意思类似：创建一个如 m_pResource（ID3D12Resource 指针描述的资源对象，并把对象句柄存储在 m_RTVHandle 里，从此 m_RTVHandle 就真正指向了一个 RV 对象
 返回到堆栈上
for (uint32_t i = 0; i < SWAP_CHAIN_BUFFER_COUNT; ++i)
 {
 ComPtr DisplayPane;
 ASSERT_SUCCEEDED(s_SwapChain1->GetBuffer(i, MY_IID_PPV_ARGS(&DisplayPlane)));
 g_DisplayPlane[i].CreateFromSwapChain(L"Primary SwapChain Buffer", DisplaPlane.Detach());
 }
 循环里每一次运行，就申请了一个 RTVHandler，然后创建了一个 RTV 对象放到这个 RTVHandler 中，这个循环一共运行了 3 次（SWAP_CHAIN_BUFFER_COUNT = 3）
往下
 太多代码还未理解。
 觉得 dx12 还是需要先把概念扔出来，而不是这里这样那那样，有了整体的概念，应该好理解很多
这里有一部分描述
 https://www.csdn.net/article/a/2016-01-04/15833516
代码里有些类似乎涉及到一些算法，搜索了些资料保存下来慢慢看
 双调排序 Bitonic Sort
一些名词
 环境光遮蔽