**4.2 CPU与GPU间的交互**

**1，命令队列**

ComPtr<ID3D12CommandQueue> mCommandQueue;

D3D12\_COMMAND\_QUEUE\_DESC queueDesc = {}

ThrowIfFailed(md3dDevice->CreateCommandQueue(&queueDesc, IID\_PPV\_ARGS(&mCommandQueue)))

IID\_PPV\_ARGS：

#define IID\_PPV\_ARGS(ppType) \_\_uuidof(\*\*(ppType)), IID\_PPV\_ARGS\_Helper(ppType)

\_\_uuidof(\*\*(ppType))：获取(\*\*(ppType))的COM接口全局唯一标识符GUID

IID\_PPV\_ARGS：将ppType强制转换为void\*\*类型

**2，命令列表**

将命令列表里的命令添加到命令队列之中：

void ID3D12CommandQueue::ExecuteCommandLists(UINT count,

ID3D12CommandList \* const \*ppCommandLists);

向命令列表中添加设置视口，清除渲染目标视图和发起绘制调用的命令：

mCommandList->RSSetViewports(1,&mScreenViewPort);

mCommandList->ClearRenderTargetView()

mCommandList->DrawIndexedInstanced(36, 1, 0, 0, 0)

在调用ID3D12CommandQueue::ExecuteCommandLists方法提交命令列表之前，一定要将调用ID3D12GraphicsCommandList::Close方法将命令列表关闭。

**3，命令分配器**

创建命令分配器：

HRESULT ID3D12Device::CreateCommandAllocator(

D3D12\_COMMAND\_LIST\_TYPE type, REFIID riid, void \*\*ppCommandAllocator);

创建命令列表：

HRESULT ID3D12Device::CreateCommandList(UINT nodeMask,

D3D12\_COMMAND\_LIST\_TYPE type, ID3D12CommandAllocator \*pCommandAllocator,

ID3D12PipelineState \*pInitialState, REFIID riid, void \*\*ppCommandList);

ID3D12GraphicsCommandList::Reset

ID3D12CommandAllocator::Reset

**4，CPU与GPU间的同步**

CPU与GPU间的同步机制

**5，资源转换**

资源屏障用D3D12\_RESOURCE\_BARRIER结构体表示。

CD3DX12\_RESOURCE\_BARRIER继承自D3D12\_RESOURCE\_BARRIER结构体，并添加了一些辅助方法，定义在d3d12x.h头文件中，这个文件不属于SDK核心部分，需要自己下载。

mCommandList->ResourceBarrier(1,

&CD3CX12\_RESOURCE\_BARRIER::Transition(CurrentBackBuffer(),D3D12\_RESOURCE\_STATE\_PRESENT, D3D12\_RESOURCE\_STATE\_RENDER\_TARGET))