**Noise 3D : shaders**

* **基于HLSL/fx的shader**

(2018.1.28)Noise3D的shader相关的工程在$(SolutionDir)shader\ 文件夹下。Noise3D用的shader现在是用Shader Model 5.0的HLSL/fx写的，毕竟接触D3D11都已经是2015的事了，现在暂时还在用2010 June的DX SDK里面的fxc.exe来编译（其实现在基于LLVM/Clang的开源HLSL　shader编译器都出来了..还没时间研究)。

* **Fxc.exe,Effects 11**

(2018.1.28)shader的主源代码就在\shader\InternalShader\里面，可以用fxc.exe编译main.fx来生成.fxo，也就是pre-compiled shader bytecode，这样子在Noise3D c++端就不需要用D3DCompile()来逐个编译shader了。Noise3D用到的shader都用effects11框架组织起来，老实说用Technique和pass去组织shader还是挺方便的，不然真的要在c++端逐个shader创建，状态也自行管理。但是用Effects11有个弊端就是，Effects11不是多线程的，要用多线程优化的话需要copy effect之类的（？不知道没研究过）。暂时还是单线程啊，先实现功能再优化吧。

基本上每个renderable object/effect都有一个对应的technique去处理，例如draw skydome/box，draw mesh，draw simple geometries，draw 2d, post processing之类的都有自己的头文件和实现文件，这些都在main.fx里面#include了。

* **Uber Shader**

(2018.1.28)暂时最复杂的可渲染对象就是网格了（还没有骨骼动画），很多特效理论上应该是自行选择开关的，例如Fog，漫反射/高光/法线/CubeMap是否使用。还有动态灯光的数量和类型应该都是比较动态的东西。把这些特效及其开关控制都集成起来有几种办法：

1.运行时动态分支(if,for的运行时判断)

2.基于uniform bool的编译时分支，在c++里面动态选择shader

3.每种effect/feature都做成一个pass，每有一种特性需要叠加到颜色就issue a draw call，用Output Merger的blending来混合颜色。

其中1，2都要做成uber shader，也就是所有需要的计算都放在一个pass里面。而3感觉draw call数会暴增，先放弃。原本的方案是1，现在加了一部分的方案2，也就是Diffuse/Specular/Normal/Environment Mapping就用2的uniform bool来分支编译，雾效灯光那些暂时还用着1的运行时动态分支。

而编译器动态分支有一种问题，就是如果有n个uniform bool，总编译分支数就是2^n，内存和占用空间我都先不管了（其实这个倒可以接受（吧？），毕竟时间比空间值钱），这2^n的代码编写其实也挺呛的，**所以在/shader/TechDefGenerator/里面放了个生成分支Techniques的definition的python脚本**，hhh这个些相对简单但又没那么简单的任务写脚本语言来实现就最舒服了。