Streamline 동작 분석

# 목록

내용

[Streamline 동작 분석 1](#_Toc196681723)

[목록 1](#_Toc196681724)

[1. 동작 분석 9](#_Toc196681725)

[1.1. main() 초기 동작 9](#_Toc196681726)

[1.1.1. 목적 9](#_Toc196681727)

[1.1.2. 동작 9](#_Toc196681728)

[1.2. SLWrapper SLWrapper::Get() 9](#_Toc196681729)

[1.2.1. 목적 9](#_Toc196681730)

[1.2.2. 동작 9](#_Toc196681731)

[1.3. bool SLWrapper::Initialize\_preDevice(nvrhi::GraphicsAPI api, const bool& checkSig, const bool& SLlog) 9](#_Toc196681732)

[1.3.1. 목적 9](#_Toc196681733)

[1.3.2. 동작 9](#_Toc196681734)

[1.4. sl::Result slInit(const Preferences &pref, uint64\_t sdkVersion) 10](#_Toc196681735)

[1.4.1. 목적 10](#_Toc196681736)

[1.4.2. 동작 10](#_Toc196681737)

[1.5. donut::app::DeviceManager\* CreateDeviceManager(nvrhi::GraphicsAPI api) – DeviceManager 객체 생성 (DX11/DX12/Valkan 통합 관리 매니저 생성) 24](#_Toc196681738)

[1.5.1. 생성 결과 25](#_Toc196681739)

[1.6. bool DeviceManager::CreateWindowDeviceAndSwapChain(const DeviceCreationParameters& params, const char \*windowTitle) – 25](#_Toc196681740)

[1.6.1. SetProcessDpiAwareness – API 인식 수준 설정 25](#_Toc196681741)

[1.6.2. m\_DeviceParams 설정 25](#_Toc196681742)

[1.6.3. CreateInstance() – Instance 생성 (DXGI Factory 생성) 25](#_Toc196681743)

[1.6.4. glfw Error Callback 함수 설정 25](#_Toc196681744)

[1.6.5. glfwDefaultWindowHints() – 이미 glfwInit() 시점에서 초기화가 적용되었으나, 재 시도 (Define을 통해 Init을 검사하는 구문 존재) 25](#_Toc196681745)

[1.6.6. glfw 기본 설정 업데이트 (\_glfw.hints 멤버변수 업데이트) 26](#_Toc196681746)

[1.6.7. glfwCreateWindow() – 실제 Window 생성 26](#_Toc196681747)

[1.6.8. glfwGetFramebufferSize() – m\_DeviceParams.backBufferWidth & Height 설정 26](#_Toc196681748)

[1.6.9. glfwSetWindowUserPointer() – 26](#_Toc196681749)

[1.6.10. glfwSetWindowPos() – 윈도우 위치 설정 26](#_Toc196681750)

[1.6.11. glfw 콜백함수 설정 26](#_Toc196681751)

[1.6.12. JoySticManager.Singleton 객체를 활용해 glfwJoystickPresent 설정 27](#_Toc196681752)

[1.6.13. DeviceManagerOverride\_DX12::CreateDevice() – 기존 DX12 방식과 유사하게 Device 생성 (Hook 적용) 27](#_Toc196681753)

[1.6.14. DeviceManagerOverride\_DX12::CreateSwapChain() – 기존 DX12 방식과 유사하게 SwapChain 생성 (Hook 적용) 27](#_Toc196681754)

[1.6.15. DeviceManagerOverride\_DX12::UpdateWindowSize() – 사이즈 재설정 27](#_Toc196681755)

[1.6.16. 27](#_Toc196681756)

[1.6.17. 4 27](#_Toc196681757)

[1.6.18. 27](#_Toc196681758)

[1.7. int glfwInit(void) – GLFW API 초기화 27](#_Toc196681759)

[1.7.1. 초기화 검사 27](#_Toc196681760)

[1.7.2. Allocate 설정 27](#_Toc196681761)

[1.7.3. Platform 선택 27](#_Toc196681762)

[1.7.4. Mutex 생성 27](#_Toc196681763)

[1.7.5. TLS 메모리 생성 (error slot & context slow) 27](#_Toc196681764)

[1.7.6. Gamepad Mapping 27](#_Toc196681765)

[1.7.7. Timer 초기화 27](#_Toc196681766)

[1.7.8. Window Hint 생성 27](#_Toc196681767)

[1.8. CreateInstanceInternal 27](#_Toc196681768)

[1.9. SLWrapper::Get().SetDevice\_nvrhi(deviceManager->GetDevice()); 27](#_Toc196681769)

[1.9.1. deviceManager의 m\_Device를 27](#_Toc196681770)

[1.10. SLWrapper::Get().Initialize\_postDevice() 27](#_Toc196681771)

[1.10.1. 반사 속성 설정 27](#_Toc196681772)

[1.11. SLWrapper::Get().UpdateFeatureAvailable(deviceManager); 27](#_Toc196681773)

[1.11.1. sl::AdapterInfo 객체 설정 진행 27](#_Toc196681774)

[1.11.2. Feature 별 기능 지원 여부 검사 27](#_Toc196681775)

[1.11.3. 28](#_Toc196681776)

[1.12. while 루프 28](#_Toc196681777)

[1.12.1. 결과 28](#_Toc196681778)

[1.12.2. DLAA 기능 작동 시 오류 존재 & 이것저것 테스트 해보면서 원래 프로젝트에 붙여보자 :) 28](#_Toc196681779)

[2. Constant 28](#_Toc196681780)

[2.1. donut::log 28](#_Toc196681781)

[2.1.1. static costexpr size\_t g\_MessageBufferSize = 4096 28](#_Toc196681782)

[2.2. sl (sl\_core\_types.h) 28](#_Toc196681783)

[2.2.1. Feature 관련 28](#_Toc196681784)

[2.3. SLWrapper 29](#_Toc196681785)

[2.3.1. static constexpr uint64\_t SDK\_VERSION = sl::kSDKVersion; 29](#_Toc196681786)

[2.3.2. static constexpr int APP\_ID = 231313132; 29](#_Toc196681787)

[2.4. sl\_version 29](#_Toc196681788)

[2.5. winnt.h 29](#_Toc196681789)

[2.5.1. #define THREAD\_BASE\_PRIORITY\_LOWRT 15 // value that gets a thread to LowRealtime-1 29](#_Toc196681790)

[2.5.2. #define THREAD\_BASE\_PRIORITY\_MAX 2 // maximum thread base priority boost 29](#_Toc196681791)

[2.5.3. #define THREAD\_BASE\_PRIORITY\_MIN (-2) // minimum thread base priority boost 29](#_Toc196681792)

[2.5.4. #define THREAD\_BASE\_PRIORITY\_IDLE (-15) // value that gets a thread to idle 29](#_Toc196681793)

[2.5.5. Priorty 관련 define 29](#_Toc196681794)

[2.5.6. 30](#_Toc196681795)

[3. Singleton 30](#_Toc196681796)

[3.1. donut::log 30](#_Toc196681797)

[3.1.1. static Severity g\_MinSeverity = Severity::Info 30](#_Toc196681798)

[3.1.2. static Callback g\_Callback = &DefaultCallback 30](#_Toc196681799)

[3.1.3. static bool g\_OutputToMessageBox = true 30](#_Toc196681800)

[3.1.4. static bool g\_OutputToDebug = true 30](#_Toc196681801)

[3.1.5. static bool g\_OutputToConsole = false 30](#_Toc196681802)

[3.1.6. static std::mutex g\_LogMutex 30](#_Toc196681803)

[3.2. main 30](#_Toc196681804)

[3.2.1. std::ofstream log\_file 30](#_Toc196681805)

[3.3. sl::ota (ota.cpp) 30](#_Toc196681806)

[3.3.1. OTA s\_ota 30](#_Toc196681807)

[3.4. sl::log (log.cpp) 30](#_Toc196681808)

[3.4.1. inline static Log\* s\_log 30](#_Toc196681809)

[3.5. sl::interposer 30](#_Toc196681810)

[3.5.1. inline static Hook\* s\_hook 30](#_Toc196681811)

[3.6. sl::plugin\_manager(pluginManager,cpp) 30](#_Toc196681812)

[3.6.1. inline static PluginManager\* s\_manager 30](#_Toc196681813)

[3.6.2. inline static PluginManagerStatus s\_status = PluginManagerStatus::eUnknown; 30](#_Toc196681814)

[3.7. sl::param (parameters.cpp) 30](#_Toc196681815)

[3.7.1. inline static Parameters\* s\_param 30](#_Toc196681816)

[3.8. sl::api::PLUGIN\_NAMESPACE (plugin.h) 30](#_Toc196681817)

[3.8.1. static PLUGIN\_CTX\* s\_ctx (PLUGIN\_CTX는 Context를 기반으로 30](#_Toc196681818)

[3.9. JoyStickManager (DeviceManager.cpp) 30](#_Toc196681819)

[3.9.1. static JoyStickManager singleton 30](#_Toc196681820)

[4. Enum 30](#_Toc196681821)

[4.1. nvrhi 30](#_Toc196681822)

[4.1.1. enum class GraphicsAPI : uint8\_t 30](#_Toc196681823)

[4.2. donut::log 30](#_Toc196681824)

[4.2.1. enum class Severity 30](#_Toc196681825)

[4.3. nvrhi::Format 31](#_Toc196681826)

[4.3.1. enum class Format : uint8\_t 31](#_Toc196681827)

[4.4. sl (sl\_core\_types.h) 32](#_Toc196681828)

[4.4.1. enum class PreferenceFlags : uint64 32](#_Toc196681829)

[4.5. sl (sl\_result.h) 32](#_Toc196681830)

[4.5.1. enum class Result 32](#_Toc196681831)

[4.5.2. 1 33](#_Toc196681832)

[4.5.3. 33](#_Toc196681833)

[5. Struct 34](#_Toc196681834)

[5.1. donut::app 34](#_Toc196681835)

[5.1.1. InstanceParameter 34](#_Toc196681836)

[5.1.2. DeviceCreationParameters : public InstanceParameter 34](#_Toc196681837)

[5.2. sl (sl\_core\_types.h) 35](#_Toc196681838)

[5.2.1. Preferences 35](#_Toc196681839)

[5.2.2. FrameToken 35](#_Toc196681840)

[5.2.3. ViewportHandle 36](#_Toc196681841)

[5.2.4. Resource 36](#_Toc196681842)

[5.3. sl (sl\_dlss.h) 37](#_Toc196681843)

[5.3.1. DLSSOptions 37](#_Toc196681844)

[5.4. sl (sl\_nis.h) 37](#_Toc196681845)

[5.4.1. NISOptions 37](#_Toc196681846)

[5.5. sl (sl\_deepdvc.h) 37](#_Toc196681847)

[5.5.1. DeepDVCOptions 37](#_Toc196681848)

[5.6. sl (sl\_dlss\_g.h) 38](#_Toc196681849)

[5.6.1. DLSSGOptions 38](#_Toc196681850)

[5.6.2. DLSSGState 38](#_Toc196681851)

[5.7. sl (sl\_reflex.h) 39](#_Toc196681852)

[5.7.1. ReflexOptions 39](#_Toc196681853)

[5.8. sl::log (log.cpp) 39](#_Toc196681854)

[5.8.1. ILog 39](#_Toc196681855)

[5.8.2. Log : ILog 40](#_Toc196681856)

[5.9. WinTrust.h 40](#_Toc196681857)

[5.9.1. WINTRUST\_FILE\_INFO 40](#_Toc196681858)

[5.10. UIData.h 40](#_Toc196681859)

[5.10.1. UIData 40](#_Toc196681860)

[6. Class 42](#_Toc196681861)

[6.1. 전역선언 (SLWrapper.h) 42](#_Toc196681862)

[6.1.1. SLWrapper 42](#_Toc196681863)

[6.2. Unknwnbase.h 43](#_Toc196681864)

[6.2.1. IID\_IUnknown 43](#_Toc196681865)

[6.3. sl::thread 43](#_Toc196681866)

[6.3.1. WorkerThread 43](#_Toc196681867)

[6.4. donut::app::DeviceManager (DeviceManager.h) 44](#_Toc196681868)

[6.4.1. DeviceManager 44](#_Toc196681869)

[6.5. DeviceManagerOverride\_DX12 (DeviceManagerOverride\_DX12.cpp) 46](#_Toc196681870)

[6.5.1. DeviceManagerOverride\_DX12 46](#_Toc196681871)

[6.6. 49](#_Toc196681872)

[7. Class 별 함수 49](#_Toc196681873)

[7.1. donut::app (ApplicationBase.cpp) 49](#_Toc196681874)

[7.1.1. nvrhi::GrapicsAPI donut::app::GetGraphicsAPIFromCommandLine(int argc, const char\* const \* argv) 49](#_Toc196681875)

[7.2. main 49](#_Toc196681876)

[7.2.1. bool ProcessCommandLine(int argc, const char\* const\* argv, donut::app::DeviceCreationParameters& deviceParams, std::string& sceneName, bool& checkSig, bool& enableSLlog) 49](#_Toc196681877)

[7.2.2. void logToFile(donut::log::Severity s, char const\* txt) 51](#_Toc196681878)

[7.3. donut::log 51](#_Toc196681879)

[7.3.1. void DefaultCallback(Severity severity, const char\* message) 51](#_Toc196681880)

[7.3.2. void debug/info/waring/error/fatal(const char\* fmt...) 52](#_Toc196681881)

[7.3.3. void message 52](#_Toc196681882)

[7.4. SLWrapper 53](#_Toc196681883)

[7.4.1. void releaseResourceCallback(sl::Resource\* resource, void\* device) 53](#_Toc196681884)

[7.4.2. void allocateResourceCallback(const sl::ResourceAllocationDesc\* resDesc, void\* device) 53](#_Toc196681885)

[7.4.3. void logFunctionCallback(sl::LogType type, const char\* msg) 54](#_Toc196681886)

[7.4.4. std::wstring GetSlInterposerDllLocation() 54](#_Toc196681887)

[7.5. sl (sl\_core\_api.h) 54](#_Toc196681888)

[7.5.1. #define SL\_ENUM\_OPERATORS\_64(T) 54](#_Toc196681889)

[7.5.2. SL\_API sl::Result slInit(const sl::Preferences &pref, uint64\_t sdkVersion = sl::kSDKVersion); 55](#_Toc196681890)

[7.6. sl.cpp 56](#_Toc196681891)

[7.6.1. void ConfigureLogOverrides(log::ILog\* log) 56](#_Toc196681892)

[7.6.2. void ConfigureLogOverridesFromRegistry(log::ILog\* log) 56](#_Toc196681893)

[7.6.3. void ConfigureLogOverridesFromEnvironment(log::ILog\* log) 57](#_Toc196681894)

[7.6.4. 57](#_Toc196681895)

[7.7. sl::ota (ota.cpp) 57](#_Toc196681896)

[7.7.1. void exec 57](#_Toc196681897)

[7.8. sl::api (plugin.h) 59](#_Toc196681898)

[7.8.1. #define SL\_PLUGIN\_DEFINE(N,V1,V2,JSON,UPDATE\_JSON\_CONFIG, PLUGIN\_NAMESPACE, PLUGIN\_CTX) 59](#_Toc196681899)

[7.9. sl::security (sl\_security.h) 60](#_Toc196681900)

[7.9.1. bool verifyEmbeddedSignature(const wchar\_t\* pathToFile) 60](#_Toc196681901)

[7.9.2. bool isSignedByNVIDIA(const wchar\_t\* pathToFile) 61](#_Toc196681902)

[7.9.3. 1 64](#_Toc196681903)

[7.9.4. 64](#_Toc196681904)

[8. 유용한 함수 64](#_Toc196681905)

[9. 참고자료 64](#_Toc196681906)

[10. 신규 함수 참고 64](#_Toc196681907)

[10.1. bloaderapi.h 64](#_Toc196681908)

[10.1.1. FreeLibrary 64](#_Toc196681909)

[10.1.2. LoadLibraryExW 64](#_Toc196681910)

[10.2. stddef.h 64](#_Toc196681911)

[10.2.1. offsetof 64](#_Toc196681912)

[10.3. 서명 관련 64](#_Toc196681913)

[10.3.1. verifying https://learn.microsoft.com/en-us/windows/win32/seccrypto/example-c-program--verifying-the-signature-of-a-pe-file 64](#_Toc196681914)

[10.3.2. WinVerifyTrust - https://learn.microsoft.com/ko-kr/windows/win32/api/wintrust/nf-wintrust-winverifytrust 64](#_Toc196681915)

[10.3.3. CryptQueryObject https://learn.microsoft.com/ko-kr/windows/win32/api/wincrypt/nf-wincrypt-cryptqueryobject 64](#_Toc196681916)

[10.3.4. CryptMsgGetParam - https://learn.microsoft.com/en-us/windows/win32/api/wincrypt/nf-wincrypt-cryptmsggetparam 64](#_Toc196681917)

[10.3.5. CryptMsgOpenToDecode - https://learn.microsoft.com/ko-kr/windows/win32/api/wincrypt/nf-wincrypt-cryptmsgopentodecode 64](#_Toc196681918)

[10.3.6. CryptMsgUpdate - https://learn.microsoft.com/ko-kr/windows/win32/api/wincrypt/nf-wincrypt-cryptmsgupdate 64](#_Toc196681919)

[10.3.7. CertOpenStore - https://learn.microsoft.com/ko-kr/windows/win32/api/wincrypt/nf-wincrypt-certopenstore 64](#_Toc196681920)

[10.3.8. CertFindCertificateInStore - https://learn.microsoft.com/ko-kr/windows/win32/api/wincrypt/nf-wincrypt-certfindcertificateinstore 64](#_Toc196681921)

[10.3.9. CryptDecodeObjectEx - https://learn.microsoft.com/ko-kr/windows/win32/api/wincrypt/nf-wincrypt-cryptdecodeobjectex 64](#_Toc196681922)

[10.3.10. CertFreeCertificateContext - https://learn.microsoft.com/ko-kr/windows/win32/api/wincrypt/nf-wincrypt-certfreecertificatecontext 64](#_Toc196681923)

[10.3.11. CertCloseStore - https://learn.microsoft.com/en-us/windows/win32/api/wincrypt/nf-wincrypt-certclosestore 64](#_Toc196681924)

[10.3.12. CryptMsgClose - https://learn.microsoft.com/ko-kr/windows/win32/api/wincrypt/nf-wincrypt-cryptmsgclose 64](#_Toc196681925)

[10.4. ShellScalingApids.h 64](#_Toc196681926)

[10.4.1. SetProcessDpiAwareness - https://learn.microsoft.com/ko-kr/windows/win32/api/shellscalingapi/nf-shellscalingapi-setprocessdpiawareness 64](#_Toc196681927)

[10.4.2. 2 65](#_Toc196681928)

[10.4.3. 65](#_Toc196681929)

[11. 기존 프로젝트와 Streamline 병합 방법 65](#_Toc196681930)

[11.1. 기존 프로젝트 정합성 유지 65](#_Toc196681931)

[11.1.1. #ifdef 구문 활용 sl.interposer 선택 로직 구현 65](#_Toc196681932)

[11.2. 싱글턴 객체 연동 65](#_Toc196681933)

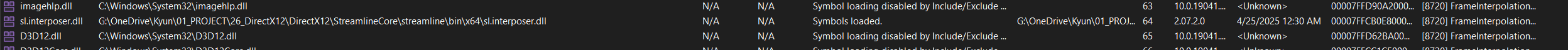
[11.2.1. sl::log 연동을 위해 dount log 관리 lib or dll 구현 65](#_Toc196681934)

[11.3. Initialize\_preDevice 구현 65](#_Toc196681935)

[11.3.1. 개선점 65](#_Toc196681936)

[11.3.2. 65](#_Toc196681937)

[11.4. 디버깅을 위해 .dll 연동 시스템 구현 (컴파일 후 \_sdk 폴더의 자료를 활용) 65](#_Toc196681938)

[11.4.1.  65](#_Toc196681939)

[11.5. Hook 시 json 연동을 위해 dll 환경변수 설정 (정합한 위치를 참조하도록 프로젝트 세팅) 65](#_Toc196681940)

[11.5.1. 결과: json 파일을 참조하여 자동 세팅 완료 65](#_Toc196681941)

[11.6. d3d12.dll 후킹 65](#_Toc196681942)

[11.6.1. 동작 65](#_Toc196681943)

[11.6.2. 결과 65](#_Toc196681944)

[11.7. 후킹 후 D3D12 셋업 66](#_Toc196681945)

[11.7.1. D3D12GetDebugInterface 66](#_Toc196681946)

[11.7.2. DXGIGetDebugInterface1 66](#_Toc196681947)

[11.7.3. CreateDXGIFactory1 66](#_Toc196681948)

[11.7.4. D3D12CreateDevice 66](#_Toc196681949)

[11.8. sl\_init 중 버그 발생 66](#_Toc196681950)

[11.8.1. 버그 발생 함수 66](#_Toc196681951)

[11.9. 개선 방법 67](#_Toc196681952)

[11.9.1. sl.\*.dll 파일에 대해 nvidia 기본 파일 활용 67](#_Toc196681953)

[11.9.2. sl.\*.dll 파일을 -debug가 아닌 -production으로 컴파일 된 파일을 적용 67](#_Toc196681954)

[11.10. D3D 객체 로드 순서 68](#_Toc196681955)

[11.10.1. DXGI Factory2 68](#_Toc196681956)

[11.11. 버그2 slSetD3DDevice 과정 중 nvngx.dll 로드 오류 68](#_Toc196681957)

[11.11.1. 에러 현상 68](#_Toc196681958)

[11.12. sl.dlss\_g.dll 라이브러리 로드 간 에러 68](#_Toc196681959)

[11.12.1. 우선 개인적으로 기본 sl.dlss\_g.dll 파일을 활용하도록 수정 68](#_Toc196681960)

[11.13. 결과 68](#_Toc196681961)

[11.13.1. 현재 2개의 swap chain을 활용하는 구조로 오류 발생. (하나의 swap chain 삭제 예정) 68](#_Toc196681962)

[11.13.2.  68](#_Toc196681963)

[11.14. Create Command Queue 68](#_Toc196681964)

[11.14.1. streamline 활용으로 수정됨에 따라 pluginManager를 활용하여 객체를 관리해야 한다. 68](#_Toc196681965)

[11.15. Create Device & Swap Chain 69](#_Toc196681966)

[11.16. Create Window 69](#_Toc196681967)

# 동작 분석

## main() 초기 동작

### 목적

#### CommandLine을 통해 전달된 옵션 적용

### 동작

#### nvrhi::GrapicsAPI 설정 (donut::app::GetGraphicsAPIFromCommandLine(\_\_argc, \_\_argv))

##### DirectX11/DirectX12/Vulkan 중 택 1

#### donut::app::DeviceCreationParameters 설정

##### backBufferWidth = 1920

##### backBufferHeight = 1080

##### swapChainSampleCount = 1

##### startFullscreen = false

##### vsyncEnable = false

##### swapChainFormat = nvrhi::format::BGRA8\_UNORM

##### enableDebugRuntime = true (단, Debug 모드이며, Vulkan API가 아닌 경우)

#### bool ProcessCommandLine()

##### donut::app::DeviceCreationParameters 설정 중 CommandLine을 통해 들어온 입력에 따라 추가 옵션 세팅

## SLWrapper SLWrapper::Get()

### 목적

#### DLSS 기능을 적용하기 위해 Streamline Function을 관리하는 Wrapper Class 연동 (Singleton으로 관리)

### 동작

#### static SLWrapper instance 반환

##### 참고사항: /Zc:threadSafeInit 옵션으로 인해 c++ 자체적으로 \_Init\_thread\_header 함수 호출 진행 (멀티쓰레드 간 오동작 방지)

## bool SLWrapper::Initialize\_preDevice(nvrhi::GraphicsAPI api, const bool& checkSig, const bool& SLlog)

### 목적

#### Singleton으로 관리되는 SLWrapper객체를 초기화

### 동작

#### 메모이제이션을 활용해 초기화 완료 검사

##### 초기화 완료 시, log::info 출력 후 true 반환

#### sl::Preferences 객체 생성

##### 로그레벨, 플러그인 경로, alloc, release, log 관련 콜백함수, option 관련 flag 관리, load 한 플러그인 관리, 어플리케이션 ID, engine, 엔진 버전, renderAPI 관리

#### pref 설정

##### allocate & release call back 함수 등록

###### allocate

DX12/DX11 버전에 따른 Resource를 생성하여 통합 sl::Resource로 반환

###### release

IUnknown 클래스로 변환한 이후, Release 함수를 호출 (COM 인터페이스를 활용하여 리소스를 관리)

##### application ID 설정

##### show Console 설정

##### log callback 함수 설정

###### Log Type에 따라 msg 출력 설정 (유용한 디버그 포인트 제공)

##### logLevel 설정

##### Load 할 Feature 설정

###### uint32\_t를 활용해 사용할 버전을 결정 (enum을 활용하는 방법도 존재하나, 초기에 구현이 이뤄진 것 같다.

###### #ifdef 구문을 활용해 로드가 필요한 구문만 실질적으로 저장시키는 방법 활용

###### 0 size 어레이가 생성되는 것을 피하기 위해 기본적으로 kFeaturePCL을 로드함

###### std::size를 활용해 배열의 크기를 저장

##### api 버전 설정 (DX12, DX11, Valkan)

##### flag 설정 (sl::PreferenceFlags)

###### 현재 flag와 bitmask 방식으로 설정을 저장 (이때, enum helpper용 define을 적용하여 구현되어있음 (enum별로 자동적으로 bit mask 함수를 제공해주는 형태)

#### DLL 관련 설정

##### 현재 path의 부모 경로의 sl.interposer.dll을 반환함

###### 참고자료

##### interposer pathDLL의 라이브러리를 load하고, 이에 대한 HMODULE interposer을 저장함

##### 로드 된 interposer가 존재하지 않는 경우, 에러 로그 출력 및 동작 종료

#### slInit(pref, SDK\_VERSION)) – 싱글턴 초기화(로그, 훅, 플러그인 매니저) 및 플러그인 호출 (1.4 참고)

#### 정상 초기화 완료 검사

## sl::Result slInit(const Preferences &pref, uint64\_t sdkVersion)

### 목적

### 동작

#### 런타임 assertion 멤버함수 offset 확인

##### sl::Preferences의 renderAPI가 138번 오프셋에 존재하는지 검사

#### **init() 람다함수 제작 (Main 중요 기능)**

##### 기본 설정

###### pref과 sdkVersion을 전달받아 활용

###### sl::Result를 반환함

##### log 싱글턴 객체 초기화

###### 스레드 초기화 (sl::thread 활용)

name: sl.log

스레드 우선순위: THREAD\_PRIORITY\_BELOW\_NORMAL (-1)

###### showConsole

###### logLevel

###### pathToLogsAndData

###### logMessageCallback

###### LogName

###### Config 설정 로드

Registry

Environment

InterposerCongig

##### Hook 싱글턴 객체 초기화

###### 인터페이스 존재 여부 검사 (존재 시 warning log 출력)

###### sl::interposer::Hook::s\_hook 싱글턴 객체 초기화 A screenshot of a computer AI-generated content may be incorrect.

##### PluginManager 싱글턴 객체 초기화

###### 멤버 변수 초기화

예시  
A screen shot of a computer

AI-generated content may be incorrect.

###### hook 함수 연동

예시  
A screenshot of a computer

AI-generated content may be incorrect.

###### m\_ota로 싱글턴 객체 연동

###### 초기화 검사

초기화가 되어있는 경우, 에러 메시지 출력 및 Result::eErrorInitNotCalled 반환

###### 멤버변수 설정

HostSDKVersion 설정

Preferences 설정

Parameter 싱글턴 객체 연동 및 플래그 설정

featureToLoad 초기화 (DLSS, DIS 등 feature 값 설정)

kFeatureCommon 값은 default로 추가

kFeaturePCL 또한 default로 추가

##### Parameter 싱글턴 객체 초기화

###### 초기화 예시

A screenshot of a computer

AI-generated content may be incorrect.

###### 콜백함수 설정

pref.allocateCallback

SLWrapper에서 설정한 allocatoe (COM 인터페이스 형태 & D3DResource)

pref.releaseCallback

SLWrapper에서 설정한 release (IUnknown으로 형변환 후 릴리즈 함수 호출)

log::getInterface

###### 결과 (m\_value에 할당이 이뤄지는 형태)

A screenshot of a computer program

AI-generated content may be incorrect.

##### sl::plugin\_manager::PluginManager::loadPlugin() – 초기 검사

###### mutex 활용 scoped\_lock 지정

###### s\_status를 메모이제이션과 같이 활용하여 초기화가 1회만 이뤄지도록 설정

##### sl::plugin\_manager::PluginManager::loadPlugin() & sl::ota::OTA::readServerManifest() – nvngx\_config.txt 내용 참조 플러그인 결정

###### C:\ProgramData/NVIDIA/NGX/models/nvngx\_config.txt 파일 참조

###### A screenshot of a computer AI-generated content may be incorrect.

##### sl::plugin\_manager::PluginManager::loadPlugin() & sl::ota::OTA::checkForOTA() - feature 별 기능 확인 후 nvngx\_update.exe 실행

###### Feature ID 지원이 가능한 apiVersion을 갖고 있는지 검사

gpu Architectre 버전 확인 (현재 제품에서는 53585, 내 RTX2070의 경우 57261)

실행 가능 시 명령어 생성

C:\Windows\System32\DriverStore\FileRepository\nv\_dispui.inf\_amd64\_6016a2508c864b42\nvngx\_update.exe -cmsid 0 -feature sl\_dlss\_0 -api update -type dll -gpuarch 0x160 -optional

thread에, execThreadProc(command) 실행 명령 후 detach

실행 결과

A black background with white and blue lines

AI-generated content may be incorrect.

스레드 형태

A screenshot of a computer

AI-generated content may be incorrect.

##### sl::plugin\_manager::PluginManager::loadPlugin() – 기존 플러그인 위치 경로 획득

###### 

###### A screenshot of a computer AI-generated content may be incorrect.

##### sl::plugin\_manager::PluginManager::loadPlugin() & sl::ota::OTA::getOTAPluginForFeature() – ngx를 활용한 플러그인 위치 경로 획득

###### ota 경로 검사 후 모델 추가 A computer screen with text and numbers AI-generated content may be incorrect.

###### 

###### 결과 A screen shot of a computer AI-generated content may be incorrect.

###### 이때, feature가 kFeatureCommon인 경우, pluginList의 0번 위치에 존재하도록 insert 함수 구현

##### sl::plugin\_manager::PluginManager::loadPlugin() & sl::plugin\_manager::PluginManager::mapPlugins - 플러그인 mapping 진행

##### sl::plugin\_manager::PluginManager::loadPlugin(const fs::path pluginFullPath, Plugin \*\*ppPlugin) – 경로에 위치하는 DLL 로드

###### 서명 확인 (13 & 14번 참고)

###### plugin 설정

fullpath

filename

lib

getFunction = slGetPluginFunction

개념: 플러그인 별 코어 API를 관리하는 함수

동작

함수명으로 요청 시 동일한 함수 반환

dlss\_dEntry.cpp 내 slGetPluginFunction 예시

#define SL\_EXPORT extern "C" \_\_declspec(dllexport)

#define SL\_EXPORT\_FUNCTION(fun)\

if (!strcmp(functionName, #fun))\

{\

    return fun;\

}

SL\_EXPORT void \*slGetPluginFunction(const char \*functionName)

{

    // Core API

    SL\_EXPORT\_FUNCTION(slOnPluginLoad);

    SL\_EXPORT\_FUNCTION(slOnPluginShutdown);

    SL\_EXPORT\_FUNCTION(slOnPluginStartup);

    SL\_EXPORT\_FUNCTION(slSetData);

    SL\_EXPORT\_FUNCTION(slGetData);

    SL\_EXPORT\_FUNCTION(slAllocateResources);

    SL\_EXPORT\_FUNCTION(slFreeResources);

    SL\_EXPORT\_FUNCTION(slIsSupported);

    SL\_EXPORT\_FUNCTION(slDLSSDSetOptions);

    SL\_EXPORT\_FUNCTION(slDLSSDGetOptimalSettings);

    SL\_EXPORT\_FUNCTION(slDLSSDGetState);

    return nullptr;

}

onLoad = slOnPluginLoad

개념:

동작

sl::PLUGIN\_NAMESPACE::s\_init을 참고하여 각 플러그인 별 싱글턴 객체 생성

파라미터로 sl::param::IParameters 설정 (모든 플러그인에서 공통된 파라미터를 설정하는 전략)

싱글턴 s\_ctx (Context)의 pluginConfigStr 문자열에 json 데이터 저장

저장된 json 데이터를 매개변수로 반환

SL\_PLUGIN\_CONTEXT\_DEFINE 예시

#define SL\_PLUGIN\_CONTEXT\_DEFINE(PLUGIN\_NAMESPACE, PLUGIN\_CTX)                               \

namespace PLUGIN\_NAMESPACE                                                                   \

{                                                                                            \

    /\* Created on DLL attached and destroyed on DLL detach from process \*/                   \

    static PLUGIN\_CTX\* s\_ctx{};                                                              \

    PLUGIN\_CTX\* getContext() { return s\_ctx; }                                               \

    static bool s\_init = false;                                                              \

}

plugin.h 예시

SL\_PLUGIN\_DEFINE("sl.dlss", Version(VERSION\_MAJOR, VERSION\_MINOR, VERSION\_PATCH), Version(0, 0, 1), JSON.c\_str(), updateEmbeddedJSON, dlss, DLSSContext)

JSON 예시 (dlss\_json.h)

unsigned char dlss\_json[] = {

123, 13, 10, 32, 32, 32, 32, 34, 105, 100, 34, 32, 58, 32, 48, 44, 13, 10, 32, 32, 32, 32, 34, 112, 114, 105, 111, 114, 105, 116, 121, 34, 32, 58, 32, 49, 48, 48, 44, 13, 10, 32, 32, 32, 32, 34, 114, 101, 113, 117, 105, 114, 101, 100, 95, 112, 108, 117, 103, 105, 110, 115, 34, 32, 58, 32, 91, 34, 115, 108, 46, 99, 111, 109, 109, 111, 110, 34, 93, 44, 13, 10, 32, 32, 32, 32, 34, 110, 97, 109, 101, 34, 32, 58, 32, 34, 115, 108, 46, 100, 108, 115, 115, 34, 44, 13, 10, 32, 32, 32, 32, 34, 110, 97, 109, 101, 115, 112, 97, 99, 101, 34, 32, 58, 32, 34, 100, 108, 115, 115, 34, 44, 13, 10, 32, 32, 32, 32, 34, 114, 104, 105, 34, 32, 58, 32, 91, 34, 100, 51, 100, 49, 49, 34, 44, 32, 34, 100, 51, 100, 49, 50, 34, 44, 32, 34, 118, 107, 34, 93, 44, 13, 10, 32, 32, 32, 32, 34, 104, 111, 111, 107, 115, 34, 32, 58, 13, 10, 32, 32, 32, 32, 91, 13, 10, 32, 32, 32, 32, 93, 13, 10, 125, 13, 10, };

unsigned int dlss\_json\_len = 205;

SL\_PLUGIN\_DEFINE 예시 (slOnPluginLoad)

//! Core definitions, each plugin must use this define and specify versions

//!

//! NOTE: This macro must be placed within 'namespace sl'

//!

#define SL\_PLUGIN\_DEFINE(N,V1,V2,JSON,UPDATE\_JSON\_CONFIG, PLUGIN\_NAMESPACE, PLUGIN\_CTX)                    \

namespace api                                                                                              \

{                                                                                                          \

    /\* Created on DLL attached and destroyed on DLL detach from process \*/                                 \

    static Context\* s\_ctx{};                                                                               \

    Context \*getContext() { return s\_ctx; }                                                                \

}                                                                                                          \

                                                                                                           \

SL\_PLUGIN\_CONTEXT\_DEFINE(PLUGIN\_NAMESPACE, PLUGIN\_CTX)                                                     \

                                                                                                           \

bool slOnPluginLoad(sl::param::IParameters \*params, const char\* loaderJSON, const char \*\*pluginJSON)       \

{                                                                                                          \

    if(!sl::PLUGIN\_NAMESPACE::s\_init)                                                                      \

    {                                                                                                      \

        sl::api::s\_ctx = new sl::api::Context(N, sl::V1, sl::V2, nullptr, nullptr,                         \

                                    new json, new json, new json);                                         \

        sl::PLUGIN\_NAMESPACE::s\_ctx = new sl::PLUGIN\_NAMESPACE::PLUGIN\_CTX();                              \

        api::s\_ctx->parameters = params;                                                                   \

        if (!plugin::onLoad(api::getContext(), loaderJSON, JSON))                                          \

        {                                                                                                  \

            return false;                                                                                  \

        }                                                                                                  \

        sl::PLUGIN\_NAMESPACE::s\_init = true;                                                               \

        json& config = \*(json\*)api::getContext()->pluginConfig;                                            \

        UPDATE\_JSON\_CONFIG(config);                                                                        \

        api::s\_ctx->pluginConfigStr = config.dump();                                                       \

    }                                                                                                      \

                                                                                                           \

    \*pluginJSON = api::s\_ctx->pluginConfigStr.c\_str();                                                     \

    return true;                                                                                           \

}                                                                                                          \

                                                                                                           \

}  /\* namespace sl \*/                                                                                      \

/\* Always in global namespace \*/                                                                           \

SL\_EXPORT BOOL APIENTRY DllMain(HMODULE hModule, DWORD fdwReason, LPVOID)                                  \

{                                                                                                          \

    switch (fdwReason)                                                                                     \

    {                                                                                                      \

        case DLL\_PROCESS\_ATTACH:                                                                           \

            break;                                                                                         \

        case DLL\_THREAD\_ATTACH:                                                                            \

            break;                                                                                         \

        case DLL\_THREAD\_DETACH:                                                                            \

            break;                                                                                         \

        case DLL\_PROCESS\_DETACH:                                                                           \

            if (!sl::PLUGIN\_NAMESPACE::s\_init) {                                                           \

                break; /\* if slOnPluginLoad() was never called, no cleanup \*/                              \

            }                                                                                              \

            delete sl::api::s\_ctx;                                                                         \

            delete sl::PLUGIN\_NAMESPACE::s\_ctx;                                                            \

            sl::api::s\_ctx = {};                                                                           \

            sl::PLUGIN\_NAMESPACE::s\_ctx = {};                                                              \

            break;                                                                                         \

    }                                                                                                      \

    return TRUE;                                                                                           \

}                                                                                                          \

namespace sl {

DLSS 구조체 예시

struct DLSSContext

{

    SL\_PLUGIN\_CONTEXT\_CREATE\_DESTROY(DLSSContext);

    void onCreateContext() {};

    void onDestroyContext() {};

    std::future<bool> initLambda;

    Constants\* commonConsts{};

    UIStats uiStats{};

    uint32\_t adapterMask{};

    common::NGXContext\* ngxContext = {};

    sl::chi::ICompute\* compute;

#ifdef SL\_CAPTURE

    sl::chi::ICapture\* capture;

#endif

    sl::chi::Kernel mvecKernel;

#ifndef SL\_PRODUCTION

    std::string ngxVersion{};

#endif

    common::PFunRegisterEvaluateCallbacks\* registerEvaluateCallbacks{};

    common::ViewportIdFrameData<4, false> constsPerViewport = { "dlss" };

    std::map<void\*, chi::ResourceState> cachedStates = {};

    std::map<void\*, NVSDK\_NGX\_Resource\_VK> cachedVkResources = {};

    std::map<uint32\_t, DLSSViewport> viewports = {};

    DLSSViewport\* viewport = {};

    RenderAPI platform;

    NVSDK\_NGX\_Resource\_VK\* cachedVkResource(sl::Resource\* res)

    {

        auto it = cachedVkResources.find(res->native);

        return it == cachedVkResources.end() ? nullptr : &(\*it).second;

    }

    void cacheState(chi::Resource res, uint32\_t nativeState = 0)

    {

        //if (cachedStates.find(res) == cachedStates.end())

        {

            // Providing state is now mandatory, defaults to "common" which is 0

            chi::ResourceState state;

            compute->getResourceState(nativeState, state);

            cachedStates[res->native] = state;

            if (res && platform == RenderAPI::eVulkan)

            {

                NVSDK\_NGX\_Resource\_VK ngx = {};

                if (res->native)

                {

                    chi::ResourceDescription desc = {};

                    desc.state = state;

                    CHI\_CHECK\_RV(compute->getResourceDescription(res, desc));

                    ngx.Resource.ImageViewInfo.ImageView = (VkImageView)res->view;

                    ngx.Resource.ImageViewInfo.Image = (VkImage)res->native;

                    sl::SubresourceRange\* subresource = findStruct<SubresourceRange>(res);

                    if (subresource)

                    {

                        ngx.Resource.ImageViewInfo.SubresourceRange = { subresource->aspectMask, subresource->baseMipLevel, subresource->levelCount, subresource->baseArrayLayer, subresource->layerCount };

                    }

                    else

                    {

                        ngx.Resource.ImageViewInfo.SubresourceRange = { VK\_IMAGE\_ASPECT\_COLOR\_BIT, 0, VK\_REMAINING\_MIP\_LEVELS, 0, VK\_REMAINING\_ARRAY\_LAYERS };

                    }

                    ngx.Resource.ImageViewInfo.Format = (VkFormat)desc.nativeFormat;

                    ngx.Resource.ImageViewInfo.Width = desc.width;

                    ngx.Resource.ImageViewInfo.Height = desc.height;

                    ngx.Type = NVSDK\_NGX\_RESOURCE\_VK\_TYPE\_VK\_IMAGEVIEW;

                    ngx.ReadWrite = (desc.flags & chi::ResourceFlags::eShaderResourceStorage) != 0;

                    cachedVkResources[res->native] = ngx;

                }

            }

        }

    }

};

onLoad 예시 (plugin.cpp)

bool onLoad(api::Context\* ctx, const char\* loaderJSON, const char\* embeddedJSON)

{

    // Setup logging and callbacks so we can report any issues correctly

    param::getPointerParam(api::getContext()->parameters, param::global::kLogInterface, &log::s\_log);

#ifndef SL\_COMMON\_PLUGIN

    param::getPointerParam(api::getContext()->parameters, param::common::kKeyboardAPI, &extra::keyboard::s\_keyboard);

#endif

    // Now let's populate our JSON config with our version and API

    json& loader = \*(json\*)ctx->loaderConfig;

    json& config = \*(json\*)ctx->pluginConfig;

    try

    {

        {

            std::istringstream stream(loaderJSON);

            stream >> loader;

        }

        if (!isLoadingAllowed(loader))

        {

            return false;

        }

        config = json::parse(embeddedJSON, nullptr, /\* allow exceptions: \*/ true, /\* ignore comments: \*/ true);

        auto pluginVersion = api::getContext()->pluginVersion;

        auto apiVersion = api::getContext()->apiVersion;

        config["version"]["major"] = pluginVersion.major;

        config["version"]["minor"] = pluginVersion.minor;

        config["version"]["build"] = pluginVersion.build;

        config["api"]["major"] = apiVersion.major;

        config["api"]["minor"] = apiVersion.minor;

        config["api"]["build"] = apiVersion.build;

        /\* If being loaded by an sl.interposer that is before version 2.3.0,

         \* then we need to enable the ABI compatibility WAR. \*/

        if (Version(loader["version"]["major"],

                    loader["version"]["minor"],

                    loader["version"]["build"]) < Version(2, 3, 0))

        {

            sl::log::g\_slEnableLogPreMetaDataUniqueWAR = true;

            SL\_LOG\_INFO("Enabling WAR for LogPreMetaDataUnique ABI Breakage");

        }

#ifndef SL\_PRODUCTION

        // Search for "sl.$(plugin\_name).json" with extra settings

        {

            const wchar\_t \*pluginPath = {};

            param::getPointerParam(ctx->parameters, param::global::kPluginPath, &pluginPath);

            std::wstring extraJSONFile = (pluginPath + std::wstring(L"/") + extra::toWStr(ctx->pluginName) + L".json").c\_str();

            if (file::exists(extraJSONFile.c\_str()))

            {

                SL\_LOG\_INFO("Found extra JSON config %S", extraJSONFile.c\_str());

                std::vector<uint8\_t> jsonText = file::read(extraJSONFile.c\_str());

                if (!jsonText.empty())

                {

                    json& extraConfig = \*(json\*)ctx->extConfig;

                    extraConfig = json::parse(jsonText.begin(), jsonText.end(), nullptr, /\* allow exceptions: \*/ true, /\* ignore comments: \*/ true);

                }

            }

        }

#endif

    }

    catch (std::exception &e)

    {

        SL\_LOG\_ERROR( "JSON exception %s", e.what());

    };

    return true;

}

context.getFunction 연동

context.isSupported 함수 연동

결과A black screen with a black text box

AI-generated content may be incorrect.

###### JSON Config 데이터 로드

populateLoaderJSON() - loader의 기본 데이터 설정

plugin->onLoad를 통해 pluginJSONText를 획득한 후, plugin 설정을 하나씩 진행

plugin 설정 결과

A screen shot of a computer

AI-generated content may be incorrect.

m\_featureExternalConfigMap - extCfg 설정

A screen shot of a computer

AI-generated content may be incorrect.

###### 현재 Load 된 plugin과 신규 플러그인 비교

###### featureToLoad 내 포함된 plugin id와 신규 plugin의 id가 일치하는지 여부 확인

일치 시 m\_featureExternalConfigMap[“feature”][“requested”] = true

###### 동일 플러그인 존재 여부 검사

동일 플러그인 존재 시, 버전을 확인하여, 더 높은 버전을 우선으로 선택

##### sl::security::verifyEmbeddedSignature – Window 서명 확인

###### wintrust.dll 로드

dll 내 WinVerifyTrust 함수를 가져온다.

###### 이후 과정은 microsoft사에서 제공하는 예제와 동일하게 동작

예제: (<https://docs.microsoft.com/en-us/windows/win32/seccrypto/example-c-program--verifying-the-signature-of-a-pe-file>)

###### 1)번 과정을 통해 가져온 WinVerifyTrust 함수를 응용해 서명 확인 진행 (dll을 통해 효율적으로 처리)

###### 서명이 되어있는 경우, NVIDIA 서명을 추가적으로 확인함

##### sl::security:: bool isSignedByNVIDIA(const wchar\_t\* pathToFile) - 서명이 정상인 경우 NVIDIA 서명 확인

###### 보안 관련 함수가 로딩되어있지 않은 경우, 필요한 함수를 로드한다.

소스코드

if (!pfnCertOpenStore)

{

    // We only support Win10+ so we can search for module in system32 directly

    auto hModCrypt32 = LoadLibraryExW(L"crypt32.dll", NULL, LOAD\_LIBRARY\_SEARCH\_SYSTEM32);

    if (!hModCrypt32 ||

        !GetProc(hModCrypt32, "CryptMsgClose", pfnCryptMsgClose) ||

        !GetProc(hModCrypt32, "CertOpenStore", pfnCertOpenStore) ||

        !GetProc(hModCrypt32, "CertCloseStore", pfnCertCloseStore) ||

        !GetProc(hModCrypt32, "CertFreeCertificateContext", pfnCertFreeCertificateContext) ||

        !GetProc(hModCrypt32, "CertFindCertificateInStore", pfnCertFindCertificateInStore) ||

        !GetProc(hModCrypt32, "CryptMsgGetParam", pfnCryptMsgGetParam) ||

        !GetProc(hModCrypt32, "CryptMsgUpdate", pfnCryptMsgUpdate) ||

        !GetProc(hModCrypt32, "CryptMsgOpenToDecode", pfnCryptMsgOpenToDecode) ||

        !GetProc(hModCrypt32, "CryptQueryObject", pfnCryptQueryObject) ||

        !GetProc(hModCrypt32, "CryptDecodeObjectEx", pfnCryptDecodeObjectEx))

    {

        return false;

    }

}

###### CryptQueryObject (암호화 API 개체의 Query 획득)

참고자료: <https://learn.microsoft.com/ko-kr/windows/win32/api/wincrypt/nf-wincrypt-cryptqueryobject> (허나, 함수 동작 간 위험성이 있음을 알리고 있음)

오류가 있는 경우 false 반환

###### 쿼리 방식

쿼리하는 개체는 파일 형태

PKCS #7 서명된 개체

반환된 개체의 예상 형식은 바이너리

###### 반환 데이터

Encoding 형식

Content 타입

포멧 형식

인증서 저장소에 대한 핸들 (동작 종료 후 CertCloseStore 함수에 전달하여 해제가 필수적이다,)

메시지의 핸들 (동작 종료 후 CryptMsgClose 함수에 전달하여 해제 필수)

###### CryptMsgGetParam (암호화 메시지 디코딩 후 메시지 매개변수를 가져온다.

2회를 읽으며, 첫번째는 메시지의 길이만을 가져온다.

LocalAlloc을 활용해 메모리 공간 할당 (<https://learn.microsoft.com/ko-kr/windows/win32/memory/global-and-local-functions>)

실제 메시지 로드

###### 메시지 내에 포함되어있는 서명 비교

동작 (현재 서명이 2개이며, 하나는 일치하는 것을 볼 수 있다)  
A screenshot of a computer program

AI-generated content may be incorrect.

###### CryptMsgOpenToDecode (디코딩을 위한 암호화 메시지를 열고 메시지의 핸들을 반환)

참고자료: <https://learn.microsoft.com/ko-kr/windows/win32/api/wincrypt/nf-wincrypt-cryptmsgopentodecode>

###### CryptMsgUpdate (암호화 메시지에 컨텐츠 추가)

암호화 메시지를 디코딩 함 (Decode를 위해 열었기 때문)

###### CryptMsgGetParam (복호화가 완료된 메시지를 읽기 위해 2회의 메시지 읽기 진행)

###### CertOpenStore (인증서 저장소 열기)

###### CertFindCertificateInStore (CERT\_CONTEXT 구조체 반환)

###### CryptDecodeObjectEx (디코딩 진행)

398개를 411개의 byte로 디코딩 완료

개수와 데이터로 반환함

###### s\_rsaStreamlinePublicKey와 Decode 된 Key가 동일한지 비교

###### LocalFree(decode Key)

###### CertFreeCertificateContext(pCertContext) (참조 수 감소)

###### CertCloseStore(hStore2, CERT\_CLOSE\_STORE\_FORCE\_FLAG) (핸들을 닫고, 저장소의 참조 횟수 감소)

###### LocalFree(pSignerInfo2)

###### CryptMsgClose(hMsg2)

###### LocalFree(pSignerInfo);

###### CryptMsgClose(hMsg);

###### CertCloseStore(hStore, CERT\_CLOSE\_STORE\_FORCE\_FLAG);

###### 결과 반환

마지막 WinTrustData도 종료 진행

##### 현재 자체 빌드한 SL DLL를 활용하는 문제로 인해, 서명이 이뤄져 있지 않은 DLL으로 로드가 이뤄지지 않은 것을 알 수 있다.

###### 참고자료: <https://github.com/NVIDIAGameWorks/Streamline/blob/main/docs/ProgrammingGuide.md> (2.1.1. 참고)

###### 이로 인해 우리는, 서명이 되어있지 않은 DLL을 참조할 수 있도록, 추후 필요 시 서명 검사 부분에 대해 예외처리를 적용해야 한다. (현재로서는 큰 변경이 없으므로, ngx 경로의 dll을 사용해도 무관)

##### std::sort - 플러그인을 우선순위에 따라 재정렬 (이때, 정렬 간 람다함수 응용)

##### 플러그인.feature.supported 검사 (지원이 되지 않는 경우, unload list에 추가)

###### 이후, 해당 name이 재 등장 시 continue로 동작 최적화

##### 결과!!!!

###### A black background with many small colored dots AI-generated content may be incorrect.

#### init() 람다함수 반환 값 리턴

## donut::app::DeviceManager\* CreateDeviceManager(nvrhi::GraphicsAPI api) – DeviceManager 객체 생성 (DX11/DX12/Valkan 통합 관리 매니저 생성)

### 생성 결과

#### A screenshot of a computer AI-generated content may be incorrect.

## bool DeviceManager::CreateWindowDeviceAndSwapChain(const DeviceCreationParameters& params, const char \*windowTitle) –

### SetProcessDpiAwareness – API 인식 수준 설정

### m\_DeviceParams 설정

### CreateInstance() – Instance 생성 (DXGI Factory 생성)

#### glfwInit() - GLFW 초기화

#### CreateInstanceInternal() – DXGI Factory 생성

### glfw Error Callback 함수 설정

### glfwDefaultWindowHints() – 이미 glfwInit() 시점에서 초기화가 적용되었으나, 재 시도 (Define을 통해 Init을 검사하는 구문 존재)

### glfw 기본 설정 업데이트 (\_glfw.hints 멤버변수 업데이트)

#### A black screen with a purple border AI-generated content may be incorrect.

#### GLFW\_RED\_BITS

#### GLFW\_GREEN\_BITS

#### GLFW\_BLUE\_BITS

#### GLFW\_ALPHA\_BITS

#### GLFW\_DEPTH\_BITS

#### GLFW\_STENCIL\_BITS

#### GLFW\_SAMPLES

#### GLFW\_REFRESH\_RATE

#### GLFW\_CLIENT\_API

#### GLFW\_VISIBLE

### glfwCreateWindow() – 실제 Window 생성

### glfwGetFramebufferSize() – m\_DeviceParams.backBufferWidth & Height 설정

### glfwSetWindowUserPointer() –

### glfwSetWindowPos() – 윈도우 위치 설정

### glfw 콜백함수 설정

glfwSetWindowPosCallback(m\_Window, WindowPosCallback\_GLFW);

glfwSetWindowCloseCallback(m\_Window, WindowCloseCallback\_GLFW);

glfwSetWindowRefreshCallback(m\_Window, WindowRefreshCallback\_GLFW);

glfwSetWindowFocusCallback(m\_Window, WindowFocusCallback\_GLFW);

glfwSetWindowIconifyCallback(m\_Window, WindowIconifyCallback\_GLFW);

glfwSetKeyCallback(m\_Window, KeyCallback\_GLFW);

glfwSetCharModsCallback(m\_Window, CharModsCallback\_GLFW);

glfwSetCursorPosCallback(m\_Window, MousePosCallback\_GLFW);

glfwSetMouseButtonCallback(m\_Window, MouseButtonCallback\_GLFW);

glfwSetScrollCallback(m\_Window, MouseScrollCallback\_GLFW);

glfwSetJoystickCallback(JoystickConnectionCallback\_GLFW);

### JoySticManager.Singleton 객체를 활용해 glfwJoystickPresent 설정

### DeviceManagerOverride\_DX12::CreateDevice() – 기존 DX12 방식과 유사하게 Device 생성 (Hook 적용)

### DeviceManagerOverride\_DX12::CreateSwapChain() – 기존 DX12 방식과 유사하게 SwapChain 생성 (Hook 적용)

### DeviceManagerOverride\_DX12::UpdateWindowSize() – 사이즈 재설정

### 

### 4

### 

## int glfwInit(void) – GLFW API 초기화

### 초기화 검사

### Allocate 설정

### Platform 선택

### Mutex 생성

### TLS 메모리 생성 (error slot & context slow)

#### 생성 간 오류 시 삭제 진행

### Gamepad Mapping

### Timer 초기화

### Window Hint 생성

## CreateInstanceInternal

## SLWrapper::Get().SetDevice\_nvrhi(deviceManager->GetDevice());

### deviceManager의 m\_Device를

## SLWrapper::Get().Initialize\_postDevice()

### 반사 속성 설정

## SLWrapper::Get().UpdateFeatureAvailable(deviceManager);

### sl::AdapterInfo 객체 설정 진행

### Feature 별 기능 지원 여부 검사

#### dlss

#### nis

#### dlss\_fg

#### reflex

#### deepdvc

#### latewarp

#### 결과

##### A screenshot of a computer program AI-generated content may be incorrect.

### 

## while 루프

### 결과

#### A screenshot of a computer screen AI-generated content may be incorrect.

### DLAA 기능 작동 시 오류 존재 & 이것저것 테스트 해보면서 원래 프로젝트에 붙여보자 :)

## DeviceManager::RunMessageLoop()

### While문 종료조건 검사

### m\_callbacks.beforeFrame이 존재하는 경우, beforeFrame 콜백함수를 beforeFrame(\*this, m\_FrameIndex)로 교체

### void glfwPollEvents() (기존의 Message 처리 부분과 동일)

#### 동작

##### 메시지 인식

##### 메시지에 따라 개별적인 처리 적용

#### 함수

void \_glfwPollEventsWin32(void)

{

    MSG msg;

    HWND handle;

    \_GLFWwindow\* window;

    while (PeekMessageW(&msg, NULL, 0, 0, PM\_REMOVE))

    {

        if (msg.message == WM\_QUIT)

        {

            // NOTE: While GLFW does not itself post WM\_QUIT, other processes

            //       may post it to this one, for example Task Manager

            // HACK: Treat WM\_QUIT as a close on all windows

            window = \_glfw.windowListHead;

            while (window)

            {

                \_glfwInputWindowCloseRequest(window);

                window = window->next;

            }

        }

        else

        {

            TranslateMessage(&msg);

            DispatchMessageW(&msg);

        }

    }

    // HACK: Release modifier keys that the system did not emit KEYUP for

    // NOTE: Shift keys on Windows tend to "stick" when both are pressed as

    //       no key up message is generated by the first key release

    // NOTE: Windows key is not reported as released by the Win+V hotkey

    //       Other Win hotkeys are handled implicitly by \_glfwInputWindowFocus

    //       because they change the input focus

    // NOTE: The other half of this is in the WM\_\*KEY\* handler in windowProc

    handle = GetActiveWindow();

    if (handle)

    {

        window = GetPropW(handle, L"GLFW");

        if (window)

        {

            int i;

            const int keys[4][2] =

            {

                { VK\_LSHIFT, GLFW\_KEY\_LEFT\_SHIFT },

                { VK\_RSHIFT, GLFW\_KEY\_RIGHT\_SHIFT },

                { VK\_LWIN, GLFW\_KEY\_LEFT\_SUPER },

                { VK\_RWIN, GLFW\_KEY\_RIGHT\_SUPER }

            };

            for (i = 0;  i < 4;  i++)

            {

                const int vk = keys[i][0];

                const int key = keys[i][1];

                const int scancode = \_glfw.win32.scancodes[key];

                if ((GetKeyState(vk) & 0x8000))

                    continue;

                if (window->keys[key] != GLFW\_PRESS)

                    continue;

                \_glfwInputKey(window, key, scancode, GLFW\_RELEASE, getKeyMods());

            }

        }

    }

    window = \_glfw.win32.disabledCursorWindow;

    if (window)

    {

        int width, height;

        \_glfwGetWindowSizeWin32(window, &width, &height);

        // NOTE: Re-center the cursor only if it has moved since the last call,

        //       to avoid breaking glfwWaitEvents with WM\_MOUSEMOVE

        // The re-center is required in order to prevent the mouse cursor stopping at the edges of the screen.

        if (window->win32.lastCursorPosX != width / 2 ||

            window->win32.lastCursorPosY != height / 2)

        {

            \_glfwSetCursorPosWin32(window, width / 2, height / 2);

        }

    }

}

### UpdateWindowSize()

#### 동작

##### width및 height에 대한 정보 획득

##### width또는 height가 0인 경우 m\_windowVisible = false를 만든 후 return

##### m\_windowVisible = true

##### glfwGetWindowAttrib()함수를 호출하여 window가 어떤 형태로 존재하는지 확인

###### 코드

GLFWAPI int glfwGetWindowAttrib(GLFWwindow\* handle, int attrib)

{

    \_GLFWwindow\* window = (\_GLFWwindow\*) handle;

    assert(window != NULL);

    \_GLFW\_REQUIRE\_INIT\_OR\_RETURN(0);

    switch (attrib)

    {

        case GLFW\_FOCUSED:

            return \_glfw.platform.windowFocused(window);

        case GLFW\_ICONIFIED:

            return \_glfw.platform.windowIconified(window);

        case GLFW\_VISIBLE:

            return \_glfw.platform.windowVisible(window);

        case GLFW\_MAXIMIZED:

            return \_glfw.platform.windowMaximized(window);

        case GLFW\_HOVERED:

            return \_glfw.platform.windowHovered(window);

        case GLFW\_FOCUS\_ON\_SHOW:

            return window->focusOnShow;

        case GLFW\_MOUSE\_PASSTHROUGH:

            return window->mousePassthrough;

        case GLFW\_TRANSPARENT\_FRAMEBUFFER:

            return \_glfw.platform.framebufferTransparent(window);

        case GLFW\_RESIZABLE:

            return window->resizable;

        case GLFW\_DECORATED:

            return window->decorated;

        case GLFW\_FLOATING:

            return window->floating;

        case GLFW\_AUTO\_ICONIFY:

            return window->autoIconify;

        case GLFW\_DOUBLEBUFFER:

            return window->doublebuffer;

        case GLFW\_CLIENT\_API:

            return window->context.client;

        case GLFW\_CONTEXT\_CREATION\_API:

            return window->context.source;

        case GLFW\_CONTEXT\_VERSION\_MAJOR:

            return window->context.major;

        case GLFW\_CONTEXT\_VERSION\_MINOR:

            return window->context.minor;

        case GLFW\_CONTEXT\_REVISION:

            return window->context.revision;

        case GLFW\_CONTEXT\_ROBUSTNESS:

            return window->context.robustness;

        case GLFW\_OPENGL\_FORWARD\_COMPAT:

            return window->context.forward;

        case GLFW\_CONTEXT\_DEBUG:

            return window->context.debug;

        case GLFW\_OPENGL\_PROFILE:

            return window->context.profile;

        case GLFW\_CONTEXT\_RELEASE\_BEHAVIOR:

            return window->context.release;

        case GLFW\_CONTEXT\_NO\_ERROR:

            return window->context.noerror;

    }

    \_glfwInputError(GLFW\_INVALID\_ENUM, "Invalid window attribute 0x%08X", attrib);

    return 0;

}

##### backBuffer의 Width와 Height를 비교하여, 이전 프레임 창의 크기와 현재 창이 동일한지 검사한다.

#### 함수

void DeviceManager::UpdateWindowSize()

{

    int width;

    int height;

    glfwGetWindowSize(m\_Window, &width, &height);

    if (width == 0 || height == 0)

    {

        // window is minimized

        m\_windowVisible = false;

        return;

    }

    m\_windowVisible = true;

    m\_windowIsInFocus = glfwGetWindowAttrib(m\_Window, GLFW\_FOCUSED) == 1;

    if (int(m\_DeviceParams.backBufferWidth) != width ||

        int(m\_DeviceParams.backBufferHeight) != height ||

        (m\_DeviceParams.vsyncEnabled != m\_RequestedVSync && GetGraphicsAPI() == nvrhi::GraphicsAPI::VULKAN))

    {

        // window is not minimized, and the size has changed

        BackBufferResizing();

        m\_DeviceParams.backBufferWidth = width;

        m\_DeviceParams.backBufferHeight = height;

        m\_DeviceParams.vsyncEnabled = m\_RequestedVSync;

        ResizeSwapChain();

        BackBufferResized();

    }

    m\_DeviceParams.vsyncEnabled = m\_RequestedVSync;

}

### AnimateRenderPresent() – 애니메이션 업데이트 및 Frame Skip 기능 추가 적용

#### 동작

##### 현재 시간과 이전 프레임의 시간을 비교하여 elapsedTime 계산

##### JoyStickManager::EraseDisconnetedJoysticks() – 삭제가 필요한 조이스틱 제거

###### 동작

RemovedJoysticks가 존재하는지 검사

삭제된 조이스틱이 있다면, 현재 프레임에서는 가장 마지막 조이스틱을 m\_JoysticksIDs 배열에서 삭제

###### 함수

void JoyStickManager::EraseDisconnectedJoysticks()

{

while (!m\_RemovedJoysticks.empty())

{

auto id = m\_RemovedJoysticks.back();

m\_RemovedJoysticks.pop\_back();

auto it = std::find(m\_JoystickIDs.begin(), m\_JoystickIDs.end(), id);

if (it != m\_JoystickIDs.end())

m\_JoystickIDs.erase(it);

}

}

##### JoyStickManager:: UpdateAllJoysticks() – 조이스틱 입력에 따라 RenderPass 정보 업데이트

###### 동작

JoyStick 입력에 따라 RenderPass 정보 업데이트

###### 함수

void JoyStickManager::UpdateAllJoysticks(const std::list<IRenderPass\*>& passes)

{

for (auto j = m\_JoystickIDs.begin(); j != m\_JoystickIDs.end(); ++j)

UpdateJoystick(\*j, passes);

}

##### DeviceManager::ShouldRenderUnfocused() - ??

##### m\_callbacks.beforeAnimate – 애니메이션 함수 호출 전 before callback 함수 검사

###### 소스코드

void SLWrapper::ReflexCallback\_SimStart(donut::app::DeviceManager& manager, uint32\_t frameID) {

    if (SLWrapper::Get().GetPCLAvailable()){

        sl::FrameToken\* temp;

        successCheck(slGetNewFrameToken(temp, &frameID), "SL\_GetFrameToken");

        successCheck(slPCLSetMarker(sl::PCLMarker::eSimulationStart, \*temp), "PCL\_SimStart");

    }

}

##### DeviceManager::Animate() – RenderPass에 포함된 Animation 관련 업데이트 (실제 객체 & Imgui)

##### m\_callbacks.afterAnimate – PCL 기능을 확인하고, **추가 동작은 추후 정리!**

###### 소스코드

void SLWrapper::ReflexCallback\_SimEnd(donut::app::DeviceManager& manager, uint32\_t frameID) {

    if (SLWrapper::Get().GetPCLAvailable())

    {

        sl::FrameToken\* temp;

        successCheck(slGetNewFrameToken(temp, &frameID), "SL\_GetFrameToken");

        successCheck(slPCLSetMarker(sl::PCLMarker::eSimulationEnd, \*temp), "PCL\_SimEnd");

    }

}

##### BeginFrame() – Streamline 기능을 핵심적으로 적용하는 파트

###### 동작

###### 소스코드

bool DeviceManagerOverride\_DX12::BeginFrame()

{

    bool turn\_on;

    // STREAMLINE

    if (SLWrapper::Get().Get\_DLSSG\_SwapChainRecreation(turn\_on)) {

        waitForQueue();

        SLWrapper::Get().CleanupDLSSG(true);

        // Get new sizes

        DXGI\_SWAP\_CHAIN\_DESC1 newSwapChainDesc;

        if (SUCCEEDED(m\_SwapChain\_native->GetDesc1(&newSwapChainDesc))) {

            m\_SwapChainDesc.Width = newSwapChainDesc.Width;

            m\_SwapChainDesc.Height = newSwapChainDesc.Height;

            m\_DeviceParams.backBufferWidth = newSwapChainDesc.Width;

            m\_DeviceParams.backBufferHeight = newSwapChainDesc.Height;

        }

        BackBufferResizing();

        // Delete swapchain and resources

        m\_SwapChain->SetFullscreenState(false, nullptr);

        ReleaseRenderTargets();

        m\_SwapChain = nullptr;

        m\_SwapChain\_native = nullptr;

        // If we turn off dlssg, then unload dlssg featuree

        if (turn\_on)

            SLWrapper::Get().FeatureLoad(sl::kFeatureDLSS\_G, true);

        else {

            SLWrapper::Get().FeatureLoad(sl::kFeatureDLSS\_G, false);

        }

        m\_UseProxySwapchain = turn\_on;

        // Recreate Swapchain and resources

        RefCountPtr<IDXGISwapChain1> pSwapChain1\_base;

        auto hr = m\_DxgiFactory2->CreateSwapChainForHwnd(m\_GraphicsQueue, m\_hWnd, &m\_SwapChainDesc, &m\_FullScreenDesc, nullptr, &pSwapChain1\_base);

        if (hr != S\_OK)  donut::log::fatal("CreateSwapChainForHwnd failed");

        hr = pSwapChain1\_base->QueryInterface(IID\_PPV\_ARGS(&m\_SwapChain));

        if (hr != S\_OK)  donut::log::fatal("QueryInterface failed");

        SLWrapper::Get().ProxyToNative(m\_SwapChain, (void\*\*)&m\_SwapChain\_native);

        if (!CreateRenderTargets())

            donut::log::fatal("CreateRenderTarget failed");

        BackBufferResized();

        // Reload DLSSG

        SLWrapper::Get().FeatureLoad(sl::kFeatureDLSS\_G, true);

        SLWrapper::Get().Quiet\_DLSSG\_SwapChainRecreation();

    }

    else if (SLWrapper::Get().Get\_Latewarp\_SwapChainRecreation(turn\_on)) {

        waitForQueue();

        // Get new sizes

        DXGI\_SWAP\_CHAIN\_DESC1 newSwapChainDesc;

        if (SUCCEEDED(m\_SwapChain\_native->GetDesc1(&newSwapChainDesc))) {

            m\_SwapChainDesc.Width = newSwapChainDesc.Width;

            m\_SwapChainDesc.Height = newSwapChainDesc.Height;

            m\_DeviceParams.backBufferWidth = newSwapChainDesc.Width;

            m\_DeviceParams.backBufferHeight = newSwapChainDesc.Height;

        }

        BackBufferResizing();

        // Delete swapchain and resources

        m\_SwapChain->SetFullscreenState(false, nullptr);

        ReleaseRenderTargets();

        m\_SwapChain = nullptr;

        m\_SwapChain\_native = nullptr;

        // If we turn off Latewarp, then unload Latewarp feature

        if (turn\_on) {

            SLWrapper::Get().FeatureLoad(sl::kFeatureLatewarp, true);

        } else {

            SLWrapper::Get().FeatureLoad(sl::kFeatureLatewarp, false);

        }

        m\_UseProxySwapchain = turn\_on;

        // Recreate Swapchain and resources

        RefCountPtr<IDXGISwapChain1> pSwapChain1\_base;

        auto hr = m\_DxgiFactory2->CreateSwapChainForHwnd(m\_GraphicsQueue, m\_hWnd, &m\_SwapChainDesc, &m\_FullScreenDesc, nullptr, &pSwapChain1\_base);

        if (hr != S\_OK)  donut::log::fatal("CreateSwapChainForHwnd failed");

        hr = pSwapChain1\_base->QueryInterface(IID\_PPV\_ARGS(&m\_SwapChain));

        if (hr != S\_OK)  donut::log::fatal("QueryInterface failed");

        SLWrapper::Get().ProxyToNative(m\_SwapChain, (void\*\*)&m\_SwapChain\_native);

        if (!CreateRenderTargets())

            donut::log::fatal("CreateRenderTarget failed");

        BackBufferResized();

        // Reload Latewarp

        SLWrapper::Get().FeatureLoad(sl::kFeatureLatewarp, true);

        SLWrapper::Get().Quiet\_Latewarp\_SwapChainRecreation();

    }

    else

    {

        DXGI\_SWAP\_CHAIN\_DESC1 newSwapChainDesc;

        DXGI\_SWAP\_CHAIN\_FULLSCREEN\_DESC newFullScreenDesc;

        if (SUCCEEDED(m\_SwapChain->GetDesc1(&newSwapChainDesc)) && SUCCEEDED(m\_SwapChain->GetFullscreenDesc(&newFullScreenDesc)))

        {

            if (m\_FullScreenDesc.Windowed != newFullScreenDesc.Windowed)

            {

                waitForQueue();

                BackBufferResizing();

                m\_FullScreenDesc = newFullScreenDesc;

                m\_SwapChainDesc = newSwapChainDesc;

                m\_DeviceParams.backBufferWidth = newSwapChainDesc.Width;

                m\_DeviceParams.backBufferHeight = newSwapChainDesc.Height;

                if (newFullScreenDesc.Windowed)

                    glfwSetWindowMonitor(m\_Window, nullptr, 50, 50, newSwapChainDesc.Width, newSwapChainDesc.Height, 0);

                ResizeSwapChain();

                BackBufferResized();

            }

        }

    }

    // STREAMLINE: hook function using proxy api object

    auto bufferIndex = m\_SwapChain->GetCurrentBackBufferIndex();

    return true;

}

##### std::this\_thread::sleep\_for(std::chrono::milliseconds(0));

##### GetDevice()->runGarbageCollection();

#### 함수

bool DeviceManager::AnimateRenderPresent()

{

    double curTime = glfwGetTime();

    double elapsedTime = curTime - m\_PreviousFrameTimestamp;

JoyStickManager::Singleton().EraseDisconnectedJoysticks();

JoyStickManager::Singleton().UpdateAllJoysticks(m\_vRenderPasses);

    if (m\_windowVisible && (m\_windowIsInFocus || ShouldRenderUnfocused()))

    {

        if (m\_callbacks.beforeAnimate) m\_callbacks.beforeAnimate(\*this, m\_FrameIndex);

        Animate(elapsedTime);

        if (m\_callbacks.afterAnimate) m\_callbacks.afterAnimate(\*this, m\_FrameIndex);

        // normal rendering           : A0    R0 P0 A1 R1 P1

        // m\_SkipRenderOnFirstFrame on: A0 A1 R0 P0 A2 R1 P1

        // m\_SkipRenderOnFirstFrame simulates multi-threaded rendering frame indices, m\_FrameIndex becomes the simulation index

        // while the local variable below becomes the render/present index, which will be different only if m\_SkipRenderOnFirstFrame is set

        if (m\_FrameIndex > 0 || !m\_SkipRenderOnFirstFrame)

        {

            if (BeginFrame())

            {

                // first time entering this loop, m\_FrameIndex is 1 for m\_SkipRenderOnFirstFrame, 0 otherwise;

                uint32\_t frameIndex = m\_FrameIndex;

                if (m\_SkipRenderOnFirstFrame)

                {

                    frameIndex--;

                }

                if (m\_callbacks.beforeRender) m\_callbacks.beforeRender(\*this, frameIndex);

                Render();

                if (m\_callbacks.afterRender) m\_callbacks.afterRender(\*this, frameIndex);

                if (m\_callbacks.beforePresent) m\_callbacks.beforePresent(\*this, frameIndex);

                bool presentSuccess = Present();

                if (m\_callbacks.afterPresent) m\_callbacks.afterPresent(\*this, frameIndex);

                if (!presentSuccess)

                {

                    return false;

                }

            }

        }

    }

    std::this\_thread::sleep\_for(std::chrono::milliseconds(0));

    GetDevice()->runGarbageCollection();

    UpdateAverageFrameTime(elapsedTime);

    m\_PreviousFrameTimestamp = curTime;

    ++m\_FrameIndex;

    return true;

}

### BeginFrame() – 프레임 처리 전 화면 크기 변화를 감지하고 Swap & Back Buffer 재조정

### m\_callbacks.beforeRender()

### Render()

### m\_callbacks.afterRender()

### m\_callbacks.beforePresent()

### Present()

### m\_callbacks.afterPresent()

### GetDevice()->waitForIdle()

## DeviceManager::Animate() – RenderPass에 포함된 Animation 업데이트 Part 1. (StreamlineSample::Animate)

### 동작

#### StreamlineSample::Animate(elapsedTime) 호출

##### FirstPersonCamera::Animate(float deltaT) 호출 – 카메라 정보 업데이트

마우스 위치 이동 정도 계산

dirty flag 및 camaraRotation 정도 초기화

마우스 좌클릭 & 마우스 이동 시

yaw & pitch 계산

cameraRotation 계산 & dirty flag 작동

FirstPersonCamera::AnimateRoll(affine3 initialRotation) 함수 호출

휠 입력시 카메라 방향에 휠 스피드만큼 회전 후 기존 구한 카메라 회전을 추가 적용

FirstPersonCamera::AnimateTranslation(float deltaT) 함수 호출

키보드 입력에 따라 카메라 위치 전환

최종적으로 획득한 카메라 회전 정보를 멤버함수에 반영

##### m\_ToneMappingPass 검사 후 AdvanceFrame 함수 호출 -> ToneMappingPasses 객체의 m\_FrameTime 정보 업데이트

##### 소요시간과 AnimationSpeed를 곱하여 기준 시간 계산

##### SceneGraphAnimationChannel::Apply(float time) - anim->Aplply(기준 시간)

###### 동작

node와 material을 weak\_ptr에서 획득

속성이 AnimationAttribute::LeafProperty인 경우 return

Sampler::Evaluate() 호출

키 프레임의 시간에 맞추서 보간 진행 (익숙한 방식) <- 이때, binary search 적용

###### 멤버변수 예제

donut::engine::SceneGraphAnimation

A screen shot of a computer

AI-generated content may be incorrect.

donut::engine::SceneGraphAnimationChannel

A screenshot of a computer

AI-generated content may be incorrect.

###### 함수

bool SceneGraphAnimationChannel::Apply(float time) const

{

    auto node = m\_TargetNode.lock();

    auto material = m\_TargetMaterial.lock();

    if ((!node && m\_Attribute != AnimationAttribute::LeafProperty) ||

        (!material && !node && m\_Attribute == AnimationAttribute::LeafProperty))

        return false;

    auto valueOption = m\_Sampler->Evaluate(time, true);

    if (!valueOption.has\_value())

        return false;

    auto value = valueOption.value();

    switch(m\_Attribute)

    {

    case AnimationAttribute::Scaling:

        node->SetScaling(dm::double3(value.xyz()));

        break;

    case AnimationAttribute::Rotation: {

        dm::dquat quat = dm::dquat::fromXYZW(dm::double4(value));

        double len = length(quat);

        if (len == 0.0)

        {

            log::warning("Rotation quaternion interpolated to zero, ignoring.");

        }

        else

        {

            quat /= len;

            node->SetRotation(quat);

        }

        break;

    }

    case AnimationAttribute::Translation:

        node->SetTranslation(dm::double3(value.xyz()));

        break;

    case AnimationAttribute::LeafProperty: {

        if (material)

        {

            if (!material->SetProperty(m\_LeafPropertyName, value))

            {

                log::warning("Cannot set property '%s' on material '%s': the material doesn't support this property.",

                    m\_LeafPropertyName.c\_str(), material->name.c\_str());

            }

        }

        else

        {

            const auto& leaf = node->GetLeaf();

            if (leaf)

            {

                if (!leaf->SetProperty(m\_LeafPropertyName, value))

                {

                    log::warning("Cannot set property '%s' on node '%s': the leaf doesn't support this property.",

                        m\_LeafPropertyName.c\_str(), node->GetName().c\_str());

                }

            }

            else

            {

                log::warning("Cannot set property '%s' on node '%s' which has no leaf.",

                    m\_LeafPropertyName.c\_str(), node->GetName().c\_str());

            }

        }

        break;

    }

    case AnimationAttribute::Undefined:

    default:

        log::warning("Unsupported animation target (%d), ignoring.", uint32\_t(m\_Attribute));

        return false;

    }

    return true;

}

#### StreamlineSample::SetLatewarpOptions() 호출

##### 함수

void StreamlineSample::SetLatewarpOptions()

{

#ifdef STREAMLINE\_FEATURE\_LATEWARP

    sl::LatewarpOptions lwOptions;

    lwOptions.latewarpActive = m\_ui.Latewarp\_active;

    SLWrapper::Get().SetLatewarpOptions(lwOptions);

#endif

    if (!m\_View || !m\_ViewPrevious || !m\_ui.Latewarp\_active)

    {

        return;

    }

    sl::ReflexCameraData cameraData{};

    std::shared\_ptr<PlanarView> planarView = std::dynamic\_pointer\_cast<PlanarView, IView>(m\_View);

    dm::affine3 viewMatrix = m\_FirstPersonCamera.GetWorldToViewMatrix();

    float verticalFov = dm::radians(m\_CameraVerticalFov);

    float2 pixelOffset = m\_ui.AAMode != AntiAliasingMode::NONE && m\_TemporalAntiAliasingPass ? m\_TemporalAntiAliasingPass->GetCurrentPixelOffset() : float2(0.f);

    float zNear = 0.01f;

    float4x4 projection = perspProjD3DStyleReverse(verticalFov, float(m\_RenderingRectSize.x) / m\_RenderingRectSize.y, zNear);

    planarView->SetViewport(nvrhi::Viewport((float) m\_RenderingRectSize.x, (float)m\_RenderingRectSize.y));

    planarView->SetPixelOffset(pixelOffset);

    planarView->SetMatrices(viewMatrix, projection);

    planarView->UpdateCache();

    cameraData.worldToViewMatrix = make\_sl\_float4x4(affineToHomogeneous(planarView->GetViewMatrix()));

    cameraData.viewToClipMatrix = make\_sl\_float4x4(planarView->GetProjectionMatrix());

    std::shared\_ptr<PlanarView> planarViewPrev = std::dynamic\_pointer\_cast<PlanarView, IView>(m\_ViewPrevious);

    cameraData.prevRenderedWorldToViewMatrix = make\_sl\_float4x4(affineToHomogeneous(planarViewPrev->GetViewMatrix()));

    cameraData.prevRenderedViewToClipMatrix = make\_sl\_float4x4(planarViewPrev->GetProjectionMatrix());

    sl::FrameToken \*frameToken = SLWrapper::Get().GetCurrentFrameToken();

    SLWrapper::Get().SetReflexCameraData(\*frameToken, cameraData);

}

### 소스코드

void DeviceManager::Animate(double elapsedTime)

{

    for(auto it : m\_vRenderPasses)

    {

        it->Animate(float(elapsedTime));

        it->SetLatewarpOptions();

    }

}

void StreamlineSample::Animate(float fElapsedTimeSeconds)

{

    m\_FirstPersonCamera.Animate(fElapsedTimeSeconds);

    if (m\_ToneMappingPass)

        m\_ToneMappingPass->AdvanceFrame(fElapsedTimeSeconds);

    if (IsSceneLoaded() && m\_ui.EnableAnimations)

    {

        m\_WallclockTime += fElapsedTimeSeconds\*m\_ui.AnimationSpeed;

        for (const auto& anim : m\_Scene->GetSceneGraph()->GetAnimations())

        {

            float duration = anim->GetDuration();

            float integral;

            float animationTime = std::modf(m\_WallclockTime / duration, &integral) \* duration;

            (void)anim->Apply(animationTime);

        }

    }

}

void FirstPersonCamera::Animate(float deltaT)

{

    // track mouse delta

    float2 mouseMove = mousePos - mousePosPrev;

    mousePosPrev = mousePos;

    bool cameraDirty = false;

    affine3 cameraRotation = affine3::identity();

    // handle mouse rotation first

    // this will affect the movement vectors in the world matrix, which we use below

    if (mouseButtonState[MouseButtons::Left] && (mouseMove.x != 0 || mouseMove.y != 0))

    {

        float yaw = m\_RotateSpeed \* mouseMove.x;

        float pitch = m\_RotateSpeed \* mouseMove.y;

        cameraRotation = rotation(float3(0.f, 1.f, 0.f), -yaw);

        cameraRotation = rotation(m\_CameraRight, -pitch) \* cameraRotation;

        cameraDirty = true;

    }

    // handle keyboard roll next

    auto rollResult = AnimateRoll(cameraRotation);

    cameraDirty |= rollResult.first;

    cameraRotation = rollResult.second;

    // handle translation

    auto translateResult = AnimateTranslation(deltaT);

    cameraDirty |= translateResult.first;

    const float3& cameraMoveVec = translateResult.second;

    if (cameraDirty)

    {

        UpdateCamera(cameraMoveVec, cameraRotation);

    }

}

std::pair<bool, affine3> FirstPersonCamera::AnimateRoll(affine3 initialRotation)

{

    bool cameraDirty = false;

    affine3 cameraRotation = initialRotation;

    if (keyboardState[KeyboardControls::RollLeft] ||

        keyboardState[KeyboardControls::RollRight])

    {

        float roll = float(keyboardState[KeyboardControls::RollLeft]) \* -m\_RotateSpeed \* 2.0f +

            float(keyboardState[KeyboardControls::RollRight]) \* m\_RotateSpeed \* 2.0f;

        cameraRotation = rotation(m\_CameraDir, roll) \* cameraRotation;

        cameraDirty = true;

    }

    return std::make\_pair(cameraDirty, cameraRotation);

}

void FirstPersonCamera::UpdateCamera(dm::float3 cameraMoveVec, dm::affine3 cameraRotation)

{

    m\_CameraPos += cameraMoveVec;

    m\_CameraDir = normalize(cameraRotation.transformVector(m\_CameraDir));

    m\_CameraUp = normalize(cameraRotation.transformVector(m\_CameraUp));

    m\_CameraRight = normalize(cross(m\_CameraDir, m\_CameraUp));

    UpdateWorldToView();

}

void BaseCamera::UpdateWorldToView()

{

    m\_MatTranslatedWorldToView = affine3::from\_cols(m\_CameraRight, m\_CameraUp, m\_CameraDir, 0.f);

    m\_MatWorldToView = translation(-m\_CameraPos) \* m\_MatTranslatedWorldToView;

}

## DeviceManager::Animate() – Part 2. (ImGui\_Renderer::Animate)

### 동작

#### 다양한 imgui 세팅 후 ImGui::NewFrame() 호출 (beginFrame())

### 소스코드

void ImGui\_Renderer::Animate(float elapsedTimeSeconds)

{

    // multiple Animate may be called before the first Render due to the m\_SkipRenderOnFirstFrame extension

    // ensure each imgui\_nvrhi->beginFrame matches with exactly one imgui\_nvrhi->Render

    if (!imgui\_nvrhi || m\_beginFrameCalled) return;

    int w, h;

    float scaleX, scaleY;

    GetDeviceManager()->GetWindowDimensions(w, h);

    GetDeviceManager()->GetDPIScaleInfo(scaleX, scaleY);

    ImGuiIO& io = ImGui::GetIO();

    io.DisplaySize = ImVec2(float(w), float(h));

    io.DisplayFramebufferScale.x = scaleX;

    io.DisplayFramebufferScale.y = scaleY;

    io.KeyCtrl = io.KeysDown[GLFW\_KEY\_LEFT\_CONTROL] || io.KeysDown[GLFW\_KEY\_RIGHT\_CONTROL];

    io.KeyShift = io.KeysDown[GLFW\_KEY\_LEFT\_SHIFT] || io.KeysDown[GLFW\_KEY\_RIGHT\_SHIFT];

    io.KeyAlt = io.KeysDown[GLFW\_KEY\_LEFT\_ALT] || io.KeysDown[GLFW\_KEY\_RIGHT\_ALT];

    io.KeySuper = io.KeysDown[GLFW\_KEY\_LEFT\_SUPER] || io.KeysDown[GLFW\_KEY\_RIGHT\_SUPER];

    imgui\_nvrhi->beginFrame(elapsedTimeSeconds);

    m\_beginFrameCalled = true;

}

## DeviceManagerOverride\_DX12::BeginFrame() – 프레임 시작 전 사이즈 재조정 등이 필요한지 검사

### 동작

#### DLSSG 기능의 SwapChain Recreation이 이뤄졌는지 확인 (RTX40 시리즈 이후 적용 가능)

#### Get\_Latewarp\_SwapChainRecreation이 이뤄졌는지 확인

#### 현재 Swap Chin Desc를 이전 프레임의 desc와 비교하여 전체화면으로 변경되었는지 검사

##### 내부 동작 예시

###### 

#### DeviceManagerOverride\_DX12::waitForQueue() – 현재 프레임 정보로 동작 완료 검사

##### m\_FrameFence를 현재 m\_FrameCount와 동기화

##### 배열로 관리하는 m\_FrameFenceEvents에 대해 m\_FrameCount에 대해 syncValue로 설정

##### 기존 - 내가 관리하던 방식

###### 

##### 소스코드

void DeviceManagerOverride\_DX12::waitForQueue() {

    auto syncValue = ++m\_FrameCount;

    m\_FrameFence->SetEventOnCompletion(syncValue, m\_FrameFenceEvents[0]);

    m\_GraphicsQueue->Signal(m\_FrameFence, syncValue);

    WaitForSingleObject(m\_FrameFenceEvents[0], INFINITE);

}

#### DeviceManager::BackBufferResizing() – Backbuffer 관련 크기를 재조정 (허나, 현 프로젝트에선 없음)

##### 동작

##### 소스코드

void DeviceManager::BackBufferResizing()

{

    m\_SwapChainFramebuffers.clear();

    for (auto it : m\_vRenderPasses)

    {

        it->BackBufferResizing();

    }

}

#### m\_FullScreenDesc, m\_SwapChainDesc, m\_DeviceParams.backBufferWidth, m\_DeviceParams.backBufferHeight를 재설정

#### ResizeSwapChain()

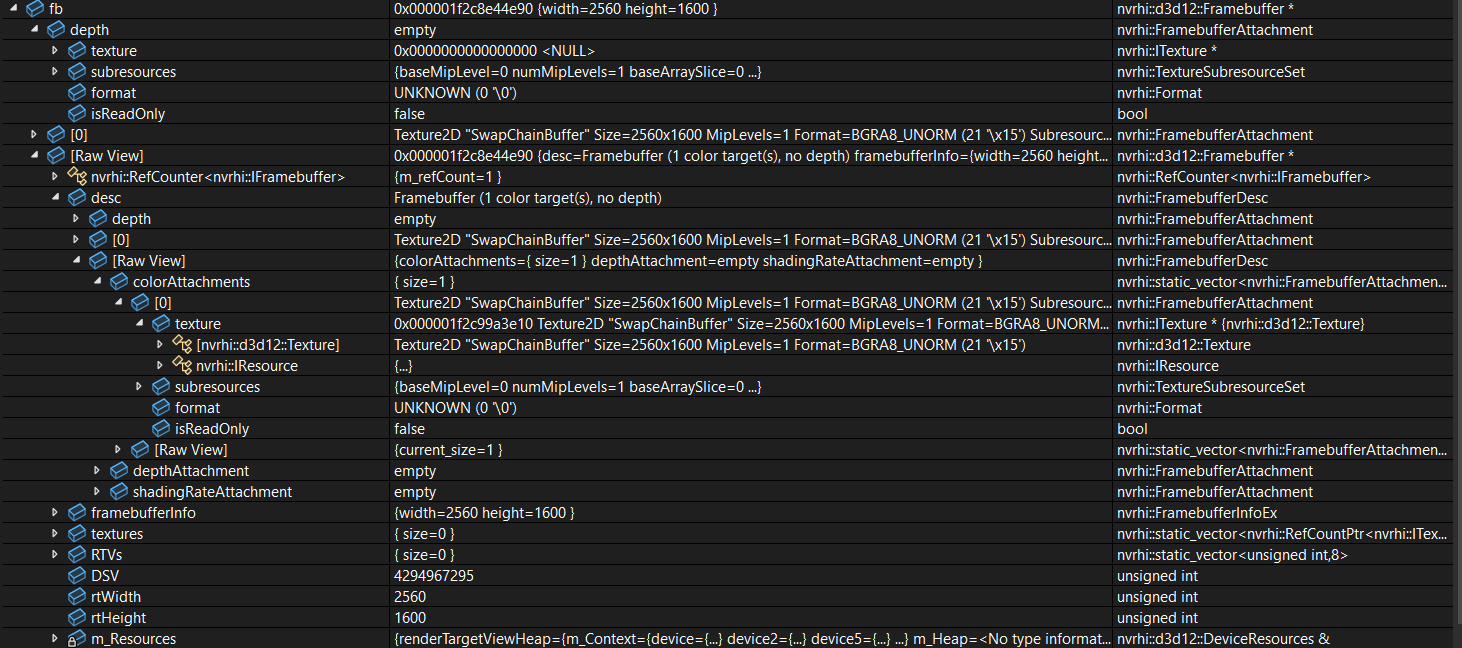
#### BackBufferResized()

##### 동작

###### Backbuffer 사이즈 조절

###### FrameBuffer 생성

class Framebuffer 데이터



##### 소스코드

###### 전문

void DeviceManager::BackBufferResized()

{

    for(auto it : m\_vRenderPasses)

    {

        it->BackBufferResized(m\_DeviceParams.backBufferWidth,

                              m\_DeviceParams.backBufferHeight,

                              m\_DeviceParams.swapChainSampleCount);

    }

    uint32\_t backBufferCount = GetBackBufferCount();

    m\_SwapChainFramebuffers.resize(backBufferCount);

    for (uint32\_t index = 0; index < backBufferCount; index++)

    {

        m\_SwapChainFramebuffers[index] = GetDevice()->createFramebuffer(

            nvrhi::FramebufferDesc().addColorAttachment(GetBackBuffer(index)));

    }

}

###### 핵심

### 소스코드

bool DeviceManagerOverride\_DX12::BeginFrame()

{

    bool turn\_on;

    // STREAMLINE

    if (SLWrapper::Get().Get\_DLSSG\_SwapChainRecreation(turn\_on)) {

        waitForQueue();

        SLWrapper::Get().CleanupDLSSG(true);

        // Get new sizes

        DXGI\_SWAP\_CHAIN\_DESC1 newSwapChainDesc;

        if (SUCCEEDED(m\_SwapChain\_native->GetDesc1(&newSwapChainDesc))) {

            m\_SwapChainDesc.Width = newSwapChainDesc.Width;

            m\_SwapChainDesc.Height = newSwapChainDesc.Height;

            m\_DeviceParams.backBufferWidth = newSwapChainDesc.Width;

            m\_DeviceParams.backBufferHeight = newSwapChainDesc.Height;

        }

        BackBufferResizing();

        // Delete swapchain and resources

        m\_SwapChain->SetFullscreenState(false, nullptr);

        ReleaseRenderTargets();

        m\_SwapChain = nullptr;

        m\_SwapChain\_native = nullptr;

        // If we turn off dlssg, then unload dlssg featuree

        if (turn\_on)

            SLWrapper::Get().FeatureLoad(sl::kFeatureDLSS\_G, true);

        else {

            SLWrapper::Get().FeatureLoad(sl::kFeatureDLSS\_G, false);

        }

        m\_UseProxySwapchain = turn\_on;

        // Recreate Swapchain and resources

        RefCountPtr<IDXGISwapChain1> pSwapChain1\_base;

        auto hr = m\_DxgiFactory2->CreateSwapChainForHwnd(m\_GraphicsQueue, m\_hWnd, &m\_SwapChainDesc, &m\_FullScreenDesc, nullptr, &pSwapChain1\_base);

        if (hr != S\_OK)  donut::log::fatal("CreateSwapChainForHwnd failed");

        hr = pSwapChain1\_base->QueryInterface(IID\_PPV\_ARGS(&m\_SwapChain));

        if (hr != S\_OK)  donut::log::fatal("QueryInterface failed");

        SLWrapper::Get().ProxyToNative(m\_SwapChain, (void\*\*)&m\_SwapChain\_native);

        if (!CreateRenderTargets())

            donut::log::fatal("CreateRenderTarget failed");

        BackBufferResized();

        // Reload DLSSG

        SLWrapper::Get().FeatureLoad(sl::kFeatureDLSS\_G, true);

        SLWrapper::Get().Quiet\_DLSSG\_SwapChainRecreation();

    }

    else if (SLWrapper::Get().Get\_Latewarp\_SwapChainRecreation(turn\_on)) {

        waitForQueue();

        // Get new sizes

        DXGI\_SWAP\_CHAIN\_DESC1 newSwapChainDesc;

        if (SUCCEEDED(m\_SwapChain\_native->GetDesc1(&newSwapChainDesc))) {

            m\_SwapChainDesc.Width = newSwapChainDesc.Width;

            m\_SwapChainDesc.Height = newSwapChainDesc.Height;

            m\_DeviceParams.backBufferWidth = newSwapChainDesc.Width;

            m\_DeviceParams.backBufferHeight = newSwapChainDesc.Height;

        }

        BackBufferResizing();

        // Delete swapchain and resources

        m\_SwapChain->SetFullscreenState(false, nullptr);

        ReleaseRenderTargets();

        m\_SwapChain = nullptr;

        m\_SwapChain\_native = nullptr;

        // If we turn off Latewarp, then unload Latewarp feature

        if (turn\_on) {

            SLWrapper::Get().FeatureLoad(sl::kFeatureLatewarp, true);

        } else {

            SLWrapper::Get().FeatureLoad(sl::kFeatureLatewarp, false);

        }

        m\_UseProxySwapchain = turn\_on;

        // Recreate Swapchain and resources

        RefCountPtr<IDXGISwapChain1> pSwapChain1\_base;

        auto hr = m\_DxgiFactory2->CreateSwapChainForHwnd(m\_GraphicsQueue, m\_hWnd, &m\_SwapChainDesc, &m\_FullScreenDesc, nullptr, &pSwapChain1\_base);

        if (hr != S\_OK)  donut::log::fatal("CreateSwapChainForHwnd failed");

        hr = pSwapChain1\_base->QueryInterface(IID\_PPV\_ARGS(&m\_SwapChain));

        if (hr != S\_OK)  donut::log::fatal("QueryInterface failed");

        SLWrapper::Get().ProxyToNative(m\_SwapChain, (void\*\*)&m\_SwapChain\_native);

        if (!CreateRenderTargets())

            donut::log::fatal("CreateRenderTarget failed");

        BackBufferResized();

        // Reload Latewarp

        SLWrapper::Get().FeatureLoad(sl::kFeatureLatewarp, true);

        SLWrapper::Get().Quiet\_Latewarp\_SwapChainRecreation();

    }

    else

    {

        DXGI\_SWAP\_CHAIN\_DESC1 newSwapChainDesc;

        DXGI\_SWAP\_CHAIN\_FULLSCREEN\_DESC newFullScreenDesc;

        if (SUCCEEDED(m\_SwapChain->GetDesc1(&newSwapChainDesc)) && SUCCEEDED(m\_SwapChain->GetFullscreenDesc(&newFullScreenDesc)))

        {

            if (m\_FullScreenDesc.Windowed != newFullScreenDesc.Windowed)

            {

                waitForQueue();

                BackBufferResizing();

                m\_FullScreenDesc = newFullScreenDesc;

                m\_SwapChainDesc = newSwapChainDesc;

                m\_DeviceParams.backBufferWidth = newSwapChainDesc.Width;

                m\_DeviceParams.backBufferHeight = newSwapChainDesc.Height;

                if (newFullScreenDesc.Windowed)

                    glfwSetWindowMonitor(m\_Window, nullptr, 50, 50, newSwapChainDesc.Width, newSwapChainDesc.Height, 0);

                ResizeSwapChain();

                BackBufferResized();

            }

        }

    }

    // STREAMLINE: hook function using proxy api object

    auto bufferIndex = m\_SwapChain->GetCurrentBackBufferIndex();

    return true;

}

# Constant

## donut::log

### static costexpr size\_t g\_MessageBufferSize = 4096

## sl (sl\_core\_types.h)

### Feature 관련

//! Features supported with this SDK

//!

//! IMPORTANT: Each feature must use a unique id

//!

using Feature = uint32\_t;

//! Deep Learning Super Sampling

constexpr Feature kFeatureDLSS = 0;

//! Real-Time Denoiser (removed)

constexpr Feature kFeatureNRD\_INVALID = 1;

//! NVIDIA Image Scaling

constexpr Feature kFeatureNIS = 2;

//! Reflex

constexpr Feature kFeatureReflex = 3;

//! PC Latency

constexpr Feature kFeaturePCL = 4;

//! DeepDVC

constexpr Feature kFeatureDeepDVC = 5;

constexpr Feature kFeatureLatewarp = 6;

//! DLSS Frame Generation

constexpr Feature kFeatureDLSS\_G = 1000;

//! DLSS Ray Reconstruction

constexpr Feature kFeatureDLSS\_RR = 1001;

constexpr Feature kFeatureNvPerf = 1002;

constexpr Feature kFeatureDirectSR = 1003;

// ImGUI

constexpr Feature kFeatureImGUI = 9999;

//! Common feature, NOT intended to be used directly

constexpr Feature kFeatureCommon = UINT\_MAX;

## SLWrapper

### static constexpr uint64\_t SDK\_VERSION = sl::kSDKVersion;

### static constexpr int APP\_ID = 231313132;

## sl\_version

## winnt.h

### #define THREAD\_BASE\_PRIORITY\_LOWRT 15 // value that gets a thread to LowRealtime-1

### #define THREAD\_BASE\_PRIORITY\_MAX 2 // maximum thread base priority boost

### #define THREAD\_BASE\_PRIORITY\_MIN (-2) // minimum thread base priority boost

### #define THREAD\_BASE\_PRIORITY\_IDLE (-15) // value that gets a thread to idle

### Priorty 관련 define

//

// Priority flags

//

#define THREAD\_PRIORITY\_LOWEST          THREAD\_BASE\_PRIORITY\_MIN

#define THREAD\_PRIORITY\_BELOW\_NORMAL    (THREAD\_PRIORITY\_LOWEST+1)

#define THREAD\_PRIORITY\_NORMAL          0

#define THREAD\_PRIORITY\_HIGHEST         THREAD\_BASE\_PRIORITY\_MAX

#define THREAD\_PRIORITY\_ABOVE\_NORMAL    (THREAD\_PRIORITY\_HIGHEST-1)

#define THREAD\_PRIORITY\_ERROR\_RETURN    (MAXLONG)

#define THREAD\_PRIORITY\_TIME\_CRITICAL   THREAD\_BASE\_PRIORITY\_LOWRT

#define THREAD\_PRIORITY\_IDLE            THREAD\_BASE\_PRIORITY\_IDLE

#define THREAD\_MODE\_BACKGROUND\_BEGIN    0x00010000

#define THREAD\_MODE\_BACKGROUND\_END      0x00020000

### 

# Singleton

## donut::log

### static Severity g\_MinSeverity = Severity::Info

### static Callback g\_Callback = &DefaultCallback

### static bool g\_OutputToMessageBox = true

### static bool g\_OutputToDebug = true

### static bool g\_OutputToConsole = false

### static std::mutex g\_LogMutex

## main

### std::ofstream log\_file

## sl::ota (ota.cpp)

### OTA s\_ota

## sl::log (log.cpp)

### inline static Log\* s\_log

## sl::interposer

### inline static Hook\* s\_hook

## sl::plugin\_manager(pluginManager,cpp)

### inline static PluginManager\* s\_manager

### inline static PluginManagerStatus s\_status = PluginManagerStatus::eUnknown;

## sl::param (parameters.cpp)

### inline static Parameters\* s\_param

## sl::api::PLUGIN\_NAMESPACE (plugin.h)

### static PLUGIN\_CTX\* s\_ctx (PLUGIN\_CTX는 Context를 기반으로

## JoyStickManager (DeviceManager.cpp)

### static JoyStickManager singleton

# Enum

## nvrhi

### enum class GraphicsAPI : uint8\_t

enum class GraphicsAPI : uint8\_t

{

    D3D11,

    D3D12,

    VULKAN

};

## donut::log

### enum class Severity

enum class Severity

{

    None = 0,

    Debug,

    Info,

    Warning,

    Error,

    Fatal

};

## nvrhi::Format

### enum class Format : uint8\_t

enum class Format : uint8\_t

{

    UNKNOWN,

    R8\_UINT,

    R8\_SINT,

    R8\_UNORM,

    R8\_SNORM,

    RG8\_UINT,

    RG8\_SINT,

    RG8\_UNORM,

    RG8\_SNORM,

    R16\_UINT,

    R16\_SINT,

    R16\_UNORM,

    R16\_SNORM,

    R16\_FLOAT,

    BGRA4\_UNORM,

    B5G6R5\_UNORM,

    B5G5R5A1\_UNORM,

    RGBA8\_UINT,

    RGBA8\_SINT,

    RGBA8\_UNORM,

    RGBA8\_SNORM,

    BGRA8\_UNORM,

    SRGBA8\_UNORM,

    SBGRA8\_UNORM,

    R10G10B10A2\_UNORM,

    R11G11B10\_FLOAT,

    RG16\_UINT,

    RG16\_SINT,

    RG16\_UNORM,

    RG16\_SNORM,

    RG16\_FLOAT,

    R32\_UINT,

    R32\_SINT,

    R32\_FLOAT,

    RGBA16\_UINT,

    RGBA16\_SINT,

    RGBA16\_FLOAT,

    RGBA16\_UNORM,

    RGBA16\_SNORM,

    RG32\_UINT,

    RG32\_SINT,

    RG32\_FLOAT,

    RGB32\_UINT,

    RGB32\_SINT,

    RGB32\_FLOAT,

    RGBA32\_UINT,

    RGBA32\_SINT,

    RGBA32\_FLOAT,

    D16,

    D24S8,

    X24G8\_UINT,

    D32,

    D32S8,

    X32G8\_UINT,

    BC1\_UNORM,

    BC1\_UNORM\_SRGB,

    BC2\_UNORM,

    BC2\_UNORM\_SRGB,

    BC3\_UNORM,

    BC3\_UNORM\_SRGB,

    BC4\_UNORM,

    BC4\_SNORM,

    BC5\_UNORM,

    BC5\_SNORM,

    BC6H\_UFLOAT,

    BC6H\_SFLOAT,

    BC7\_UNORM,

    BC7\_UNORM\_SRGB,

    COUNT,

};

## sl (sl\_core\_types.h)

### enum class PreferenceFlags : uint64

//! Optional flags

enum class PreferenceFlags : uint64\_t

{

    //! Set by default - Disables command list state tracking - Host application is responsible for restoring CL state correctly after each 'slEvaluateFeature' call

    eDisableCLStateTracking = 1 << 0,

    //! Optional - Disables debug text on screen in development builds

    eDisableDebugText = 1 << 1,

    //! Optional - IMPORTANT: Only to be used in the advanced integration mode, see the 'manual hooking' programming guide for more details

    eUseManualHooking = 1 << 2,

    //! Optional - Enables downloading of Over The Air (OTA) updates for SL and NGX

    //! This will invoke the OTA updater to look for new updates. A separate

    //! flag below is used to control whether or not OTA-downloaded SL Plugins are

    //! loaded.

    eAllowOTA = 1 << 3,

    //! Do not check OS version when deciding if feature is supported or not

    //!

    //! IMPORTANT: ONLY SET THIS FLAG IF YOU KNOW WHAT YOU ARE DOING.

    //!

    //! VARIOUS WIN APIs INCLUDING BUT NOT LIMITED TO `IsWindowsXXX`, `GetVersionX`, `rtlGetVersion` ARE KNOWN FOR RETURNING INCORRECT RESULTS.

    eBypassOSVersionCheck = 1 << 4,

    //! Optional - If specified SL will create DXGI factory proxy rather than modifying the v-table for the base interface.

    //!

    //! This can help with 3rd party overlays which are NOT integrated with the host application but rather operate via injection.

    eUseDXGIFactoryProxy = 1 << 5,

    //! Optional - Enables loading of plugins downloaded Over The Air (OTA), to

    //! be used in conjunction with the eAllowOTA flag.

    eLoadDownloadedPlugins = 1 << 6,

};

## sl (sl\_result.h)

### enum class Result

enum class Result

{

    eOk,

    eErrorIO,

    eErrorDriverOutOfDate,

    eErrorOSOutOfDate,

    eErrorOSDisabledHWS,

    eErrorDeviceNotCreated,

    eErrorNoSupportedAdapterFound,

    eErrorAdapterNotSupported,

    eErrorNoPlugins,

    eErrorVulkanAPI,

    eErrorDXGIAPI,

    eErrorD3DAPI,

    // NRD was removed

    eErrorNRDAPI,

    eErrorNVAPI,

    eErrorReflexAPI,

    eErrorNGXFailed,

    eErrorJSONParsing,

    eErrorMissingProxy,

    eErrorMissingResourceState,

    eErrorInvalidIntegration,

    eErrorMissingInputParameter,

    eErrorNotInitialized,

    eErrorComputeFailed,

    eErrorInitNotCalled,

    eErrorExceptionHandler,

    eErrorInvalidParameter,

    eErrorMissingConstants,

    eErrorDuplicatedConstants,

    eErrorMissingOrInvalidAPI,

    eErrorCommonConstantsMissing,

    eErrorUnsupportedInterface,

    eErrorFeatureMissing,

    eErrorFeatureNotSupported,

    eErrorFeatureMissingHooks,

    eErrorFeatureFailedToLoad,

    eErrorFeatureWrongPriority,

    eErrorFeatureMissingDependency,

    eErrorFeatureManagerInvalidState,

    eErrorInvalidState,

    eWarnOutOfVRAM,

};

### 1

### 

# Struct

## donut::app

### InstanceParameter

    struct InstanceParameters

    {

        bool enableDebugRuntime = false;

        bool enableGPUValidation = false; // Affects only DX12

        bool headlessDevice = false;

#if DONUT\_WITH\_AFTERMATH

        bool enableAftermath = false;

#endif

        // Severity of the information log messages from the device manager, like the device name or enabled extensions.

        log::Severity infoLogSeverity = log::Severity::Info;

#if DONUT\_WITH\_VULKAN

        std::vector<std::string> requiredVulkanInstanceExtensions;

        std::vector<std::string> requiredVulkanLayers;

        std::vector<std::string> optionalVulkanInstanceExtensions;

        std::vector<std::string> optionalVulkanLayers;

#endif

    };

### DeviceCreationParameters : public InstanceParameter

#### - 소스코드

    struct DeviceCreationParameters : public InstanceParameters

    {

        bool startMaximized = false;

        bool startFullscreen = false;

        bool allowModeSwitch = true;

        int windowPosX = -1;            // -1 means use default placement

        int windowPosY = -1;

        uint32\_t backBufferWidth = 1280;

        uint32\_t backBufferHeight = 720;

        uint32\_t refreshRate = 0;

        uint32\_t swapChainBufferCount = 3;

        nvrhi::Format swapChainFormat = nvrhi::Format::SRGBA8\_UNORM;

        uint32\_t swapChainSampleCount = 1;

        uint32\_t swapChainSampleQuality = 0;

        uint32\_t maxFramesInFlight = 2;

        bool enableNvrhiValidationLayer = false;

        bool vsyncEnabled = false;

        bool enableRayTracingExtensions = false; // for vulkan

        bool enableComputeQueue = false;

        bool enableCopyQueue = false;

        // Index of the adapter (DX11, DX12) or physical device (Vk) on which to initialize the device.

        // Negative values mean automatic detection.

        // The order of indices matches that returned by DeviceManager::EnumerateAdapters.

        int adapterIndex = -1;

        // set to true to enable DPI scale factors to be computed per monitor

        // this will keep the on-screen window size in pixels constant

        //

        // if set to false, the DPI scale factors will be constant but the system

        // may scale the contents of the window based on DPI

        //

        // note that the backbuffer size is never updated automatically; if the app

        // wishes to scale up rendering based on DPI, then it must set this to true

        // and respond to DPI scale factor changes by resizing the backbuffer explicitly

        bool enablePerMonitorDPI = false;

#if DONUT\_WITH\_DX11 || DONUT\_WITH\_DX12

        DXGI\_USAGE swapChainUsage = DXGI\_USAGE\_SHADER\_INPUT | DXGI\_USAGE\_RENDER\_TARGET\_OUTPUT;

        D3D\_FEATURE\_LEVEL featureLevel = D3D\_FEATURE\_LEVEL\_11\_1;

#endif

#if DONUT\_WITH\_VULKAN

        std::vector<std::string> requiredVulkanDeviceExtensions;

        std::vector<std::string> optionalVulkanDeviceExtensions;

        std::vector<size\_t> ignoredVulkanValidationMessageLocations;

        std::function<void(VkDeviceCreateInfo&)> deviceCreateInfoCallback;

        // This pointer specifies an optional structure to be put at the end of the chain for 'vkGetPhysicalDeviceFeatures2' call.

        // The structure may also be a chain, and must be alive during the device initialization process.

        // The elements of this structure will be populated before 'deviceCreateInfoCallback' is called,

        // thereby allowing applications to determine if certain features may be enabled on the device.

        void\* physicalDeviceFeatures2Extensions = nullptr;

#endif

    };

## sl (sl\_core\_types.h)

### Preferences

#### 목적

#### 소스코드

//! Application preferences

//!

//! {1CA10965-BF8E-432B-8DA1-6716D879FB14}

SL\_STRUCT\_BEGIN(Preferences, StructType({ 0x1ca10965, 0xbf8e, 0x432b, { 0x8d, 0xa1, 0x67, 0x16, 0xd8, 0x79, 0xfb, 0x14 } }), kStructVersion1)

    //! Optional - In non-production builds it is useful to enable debugging console window

    bool showConsole = false;

    //! Optional - Various logging levels

    LogLevel logLevel = LogLevel::eDefault;

    //! Optional - Absolute paths to locations where to look for plugins, first path in the list has the highest priority

    const wchar\_t\*\* pathsToPlugins{};

    //! Optional - Number of paths to search

    uint32\_t numPathsToPlugins = 0;

    //! Optional - Absolute path to location where logs and other data should be stored

    //!

    //! NOTE: Set this to nullptr in order to disable logging to a file

    const wchar\_t\* pathToLogsAndData{};

    //! Optional - Allows resource allocation tracking on the host side

    PFun\_ResourceAllocateCallback\* allocateCallback{};

    //! Optional - Allows resource deallocation tracking on the host side

    PFun\_ResourceReleaseCallback\* releaseCallback{};

    //! Optional - Allows log message tracking including critical errors if they occur

    PFun\_LogMessageCallback\* logMessageCallback{};

    //! Optional - Flags used to enable or disable advanced options

    PreferenceFlags flags = PreferenceFlags::eDisableCLStateTracking | PreferenceFlags::eAllowOTA | PreferenceFlags::eLoadDownloadedPlugins;

    //! Required - Features to load (assuming appropriate plugins are found), if not specified NO features will be loaded by default

    const Feature\* featuresToLoad{};

    //! Required - Number of features to load, only used when list is not a null pointer

    uint32\_t numFeaturesToLoad{};

    //! Optional - Id provided by NVIDIA, if not specified then engine type and version are required

    uint32\_t applicationId{};

    //! Optional - Type of the rendering engine used, if not specified then applicationId is required

    EngineType engine = EngineType::eCustom;

    //! Optional - Version of the rendering engine used

    const char\* engineVersion{};

    //! Optional - GUID (like for example 'a0f57b54-1daf-4934-90ae-c4035c19df04')

    const char\* projectId{};

    //! Optional - Which rendering API host is planning to use

    //!

    //! NOTE: To ensure correct `slGetFeatureRequirements` behavior please specify if planning to use Vulkan.

    RenderAPI renderAPI = RenderAPI::eD3D12;

    //! IMPORTANT: New members go here or if optional can be chained in a new struct, see sl\_struct.h for details

SL\_STRUCT\_END()

### FrameToken

#### 목적

#### 소스코드

//! Frame tracking handle

//!

//! IMPORTANT: Use slGetNewFrameToken to obtain unique instance

//!

//! {830A0F35-DB84-4171-A804-59B206499B18}

SL\_STRUCT\_PROTECTED\_BEGIN(FrameToken, StructType({ 0x830a0f35, 0xdb84, 0x4171, { 0xa8, 0x4, 0x59, 0xb2, 0x6, 0x49, 0x9b, 0x18 } }), kStructVersion1)

    //! Helper operator to obtain current frame index

    virtual operator uint32\_t() const = 0;

SL\_STRUCT\_END()

### ViewportHandle

#### 목적

#### 소스코드

//! Handle for the unique viewport

//!

//! {171B6435-9B3C-4FC8-9994-FBE52569AAA4}

SL\_STRUCT\_BEGIN(ViewportHandle, StructType({ 0x171b6435, 0x9b3c, 0x4fc8, { 0x99, 0x94, 0xfb, 0xe5, 0x25, 0x69, 0xaa, 0xa4 } }), kStructVersion1)

    ViewportHandle(uint32\_t v) : BaseStructure(ViewportHandle::s\_structType, kStructVersion1), value(v) {}

    ViewportHandle(int32\_t v) : BaseStructure(ViewportHandle::s\_structType, kStructVersion1), value(v) {}

    operator uint32\_t() const { return value; }

private:

    uint32\_t value = UINT\_MAX;

SL\_STRUCT\_END()

### Resource

#### 소스코드

//! Native resource

//!

//! {3A9D70CF-2418-4B72-8391-13F8721C7261}

SL\_STRUCT\_BEGIN(Resource, StructType({ 0x3a9d70cf, 0x2418, 0x4b72, { 0x83, 0x91, 0x13, 0xf8, 0x72, 0x1c, 0x72, 0x61 } }), kStructVersion1)

    //! Constructors

    //!

    //! Resource type, native pointer are MANDATORY always

    //! Resource state is MANDATORY unless using D3D11

    //! Resource view, description etc. are MANDATORY only when using Vulkan

    //!

    Resource(ResourceType \_type, void\* \_native, void\* \_mem, void\* \_view, uint32\_t \_state = UINT\_MAX) : BaseStructure(Resource::s\_structType, kStructVersion1), type(\_type), native(\_native), memory(\_mem), view(\_view), state(\_state){};

    Resource(ResourceType \_type, void\* \_native, uint32\_t \_state = UINT\_MAX) : BaseStructure(Resource::s\_structType, kStructVersion1), type(\_type), native(\_native), state(\_state) {};

    //! Conversion helpers for D3D

    inline operator ID3D12Resource\* () { return reinterpret\_cast<ID3D12Resource\*>(native); }

    inline operator ID3D11Resource\* () { return reinterpret\_cast<ID3D11Resource\*>(native); }

    inline operator ID3D11Buffer\* () { return reinterpret\_cast<ID3D11Buffer\*>(native); }

    inline operator ID3D11Texture2D\* () { return reinterpret\_cast<ID3D11Texture2D\*>(native); }

    //! Indicates the type of resource

    ResourceType type = ResourceType::eTex2d;

    //! ID3D11Resource/ID3D12Resource/VkBuffer/VkImage

    void\* native{};

    //! vkDeviceMemory or nullptr

    void\* memory{};

    //! VkImageView/VkBufferView or nullptr

    void\* view{};

    //! State as D3D12\_RESOURCE\_STATES or VkImageLayout

    //!

    //! IMPORTANT: State is MANDATORY and needs to be correct when tagged resources are actually used.

    //!

    uint32\_t state = UINT\_MAX;

    //! Width in pixels

    uint32\_t width{};

    //! Height in pixels

    uint32\_t height{};

    //! Native format

    uint32\_t nativeFormat{};

    //! Number of mip-map levels

    uint32\_t mipLevels{};

    //! Number of arrays

    uint32\_t arrayLayers{};

    //! Virtual address on GPU (if applicable)

    uint64\_t gpuVirtualAddress{};

    //! VkImageCreateFlags

    uint32\_t flags;

    //! VkImageUsageFlags

    uint32\_t usage{};

    //! Reserved for internal use

    uint32\_t reserved{};

    //! IMPORTANT: New members go here or if optional can be chained in a new struct, see sl\_struct.h for details

SL\_STRUCT\_END()

## sl (sl\_dlss.h)

### DLSSOptions

#### 목적

#### 소스코드

// {6AC826E4-4C61-4101-A92D-638D421057B8}

SL\_STRUCT\_BEGIN(DLSSOptions, StructType({ 0x6ac826e4, 0x4c61, 0x4101, { 0xa9, 0x2d, 0x63, 0x8d, 0x42, 0x10, 0x57, 0xb8 } }), kStructVersion3)

    //! Specifies which mode should be used

    DLSSMode mode = DLSSMode::eOff;

    //! Specifies output (final) target width

    uint32\_t outputWidth = INVALID\_UINT;

    //! Specifies output (final) target height

    uint32\_t outputHeight = INVALID\_UINT;

    //! Specifies sharpening level in range [0,1]

    float sharpness = 0.0f;

    //! Specifies pre-exposure value

    float preExposure = 1.0f;

    //! Specifies exposure scale value

    float exposureScale = 1.0f;

    //! Specifies if tagged color buffers are full HDR or not (DLSS in HDR pipeline or not)

    Boolean colorBuffersHDR = Boolean::eTrue;

    //! Specifies if indicator on screen should invert axis

    Boolean indicatorInvertAxisX = Boolean::eFalse;

    //! Specifies if indicator on screen should invert axis

    Boolean indicatorInvertAxisY = Boolean::eFalse;

    //! Presets

    DLSSPreset dlaaPreset = DLSSPreset::eDefault;

    DLSSPreset qualityPreset = DLSSPreset::eDefault;

    DLSSPreset balancedPreset = DLSSPreset::eDefault;

    DLSSPreset performancePreset = DLSSPreset::eDefault;

    DLSSPreset ultraPerformancePreset = DLSSPreset::eDefault;

    DLSSPreset ultraQualityPreset = DLSSPreset::eDefault;

    //! Specifies if the setting for AutoExposure is used

    Boolean useAutoExposure = Boolean::eFalse;

    //! Whether or not the alpha channel should be upscaled (if false, only RGB is upscaled)

    //! Enabling alpha upscaling may impact performance

    Boolean alphaUpscalingEnabled = Boolean::eFalse;

    //! IMPORTANT: New members go here or if optional can be chained in a new struct, see sl\_struct.h for details

SL\_STRUCT\_END()

## sl (sl\_nis.h)

### NISOptions

#### 목적

#### 소스코드

// {676610E5-9674-4D3A-9C8A-F495D01B36F3}

SL\_STRUCT\_BEGIN(NISOptions, StructType({ 0x676610e5, 0x9674, 0x4d3a, { 0x9c, 0x8a, 0xf4, 0x95, 0xd0, 0x1b, 0x36, 0xf3 } }), kStructVersion1)

    //! Specifies which mode should be used

    NISMode mode = NISMode::eScaler;

    //! Specifies which hdr mode should be used

    NISHDR hdrMode = NISHDR::eNone;

    //! Specifies sharpening level in range [0,1]

    float sharpness = 0.0f;

    //! IMPORTANT: New members go here or if optional can be chained in a new struct, see sl\_struct.h for details

SL\_STRUCT\_END()

## sl (sl\_deepdvc.h)

### DeepDVCOptions

#### 목적

#### 소스코드

// {23288AAD-7E7E-BE2A-916F-27DA30A3046B}

SL\_STRUCT\_BEGIN(DeepDVCOptions, StructType({ 0x23288aad, 0x7e7e, 0xbe2a, { 0x91, 0x67, 0x27, 0xda, 0x30, 0xa3, 0x04, 0x6b } }), kStructVersion1)

    //! Specifies which mode should be used

    DeepDVCMode mode = DeepDVCMode::eOff;

    //! Specifies intensity level in range [0,1]. Default 0.5

    float intensity = 0.5f;

    //! Specifies saturation boost in range [0,1]. Default 0.25

    float saturationBoost = 0.25f;

SL\_STRUCT\_END()

## sl (sl\_dlss\_g.h)

### DLSSGOptions

#### 목적

#### 소스코드

// {FAC5F1CB-2DFD-4F36-A1E6-3A9E865256C5}

SL\_STRUCT\_BEGIN(DLSSGOptions, StructType({ 0xfac5f1cb, 0x2dfd, 0x4f36, { 0xa1, 0xe6, 0x3a, 0x9e, 0x86, 0x52, 0x56, 0xc5 } }), kStructVersion3)

    //! Specifies which mode should be used.

    DLSSGMode mode = DLSSGMode::eOff;

    //! Number of frames to generate inbetween fully rendered frames. Cannot exceed DLSSGState::numFramesToGenerateMax.

    //!     For 2x frame multiplier, numFramesToGenerate is 1.

    //!     For 3x frame multiplier, numFramesToGenerate is 2.

    //!     For 4x frame multiplier, numFramesToGenerate is 3.

    uint32\_t numFramesToGenerate = 1;

    //! Optional - Flags used to enable or disable certain functionality

    DLSSGFlags flags{};

    //! Optional - Dynamic resolution optimal width (used only if eDynamicResolutionEnabled is set)

    uint32\_t dynamicResWidth{};

    //! Optional - Dynamic resolution optimal height (used only if eDynamicResolutionEnabled is set)

    uint32\_t dynamicResHeight{};

    //! Optional - Expected number of buffers in the swap-chain

    uint32\_t numBackBuffers{};

    //! Optional - Expected width of the input render targets (depth, motion-vector buffers etc)

    uint32\_t mvecDepthWidth{};

    //! Optional - Expected height of the input render targets (depth, motion-vector buffers etc)

    uint32\_t mvecDepthHeight{};

    //! Optional - Expected width of the back buffers in the swap-chain

    uint32\_t colorWidth{};

    //! Optional - Expected height of the back buffers in the swap-chain

    uint32\_t colorHeight{};

    //! Optional - Indicates native format used for the swap-chain back buffers

    uint32\_t colorBufferFormat{};

    //! Optional - Indicates native format used for eMotionVectors

    uint32\_t mvecBufferFormat{};

    //! Optional - Indicates native format used for eDepth

    uint32\_t depthBufferFormat{};

    //! Optional - Indicates native format used for eHUDLessColor

    uint32\_t hudLessBufferFormat{};

    //! Optional - Indicates native format used for eUIColorAndAlpha

    uint32\_t uiBufferFormat{};

    //! Optional - if specified DLSSG will return any errors which occur when calling underlying API (DXGI or Vulkan)

    PFunOnAPIErrorCallback\* onErrorCallback{};

    // kStructVersion2

    Boolean bReserved15 = eInvalid;

    // kStructVersion3

    //! Optional - determines the level of client and DLSSG queue parallelism to use for performance gain - must be same for all viewports.

    DLSSGQueueParallelismMode queueParallelismMode{};

    //! IMPORTANT: New members go here or if optional can be chained in a new struct, see sl\_struct.h for details

SL\_STRUCT\_END()

### DLSSGState

#### 목적

#### 소스코드

// {CC8AC8E1-A179-44F5-97FA-E74112F9BC61}

SL\_STRUCT\_BEGIN(DLSSGState, StructType({ 0xcc8ac8e1, 0xa179, 0x44f5, { 0x97, 0xfa, 0xe7, 0x41, 0x12, 0xf9, 0xbc, 0x61 } }), kStructVersion3)

    //! Specifies the amount of memory expected to be used

    uint64\_t estimatedVRAMUsageInBytes{};

    //! Specifies current status of DLSS-G

    DLSSGStatus status{};

    //! Specifies minimum supported dimension

    uint32\_t minWidthOrHeight{};

    //! Number of frames presented since the last 'slDLSSGGetState' call

    uint32\_t numFramesActuallyPresented{};

    // kStructVersion2

    //! Maximum number of frames possible to generate on this gpu architecture.

    //!     For 2x only supporting devices, numFramesToGenerateMax is 1.

    //!     For 3x and 4x supporting devices, numFramesToGenerateMax is 3.

    uint32\_t numFramesToGenerateMax{};

    sl::Boolean bReserved4{};

    //! Hint to the application to display VSync support in the user interface

    sl::Boolean bIsVsyncSupportAvailable{};

    //! SL client must wait on SL DLSS-G plugin-internal fence and associated value, before it can modify or destroy the tagged resources input

    //! to DLSS-G enabled for the corresponding previously presented frame on a non-presenting queue.

    //! If modified on client's presenting queue, then it's recommended but not required.

    //! However, if DLSSGQueueParallelismMode::eBlockNoClientQueues is set, then it's always required.

    //! It must call slDLSSGGetState on the present thread to retrieve the fence value for the inputs consumed by FG, on which client would

    //! wait in the frame it would modify those inputs.

    void\* inputsProcessingCompletionFence{};

    uint64\_t lastPresentInputsProcessingCompletionFenceValue{};

    //! IMPORTANT: New members go here or if optional can be chained in a new struct, see sl\_struct.h for details

SL\_STRUCT\_END()

## sl (sl\_reflex.h)

### ReflexOptions

#### 목적

#### 소스코드

// {F03AF81A-6D0B-4902-A651-C4965E215434}

SL\_STRUCT\_BEGIN(ReflexOptions, StructType({ 0xf03af81a, 0x6d0b, 0x4902, { 0xa6, 0x51, 0xc4, 0x96, 0x5e, 0x21, 0x54, 0x34 } }), kStructVersion1)

    //! Specifies which mode should be used

    ReflexMode mode = ReflexMode::eOff;

    //! Specifies if frame limiting (FPS cap) is enabled (0 to disable, microseconds otherwise).

    //! One benefit of using Reflex's FPS cap over other implementations is the driver would be aware and can provide better optimizations.

    //! This setting is independent of ReflexOptions::mode; it can even be used with mode == ReflexMode::eOff.

    //! The value is used each time you call slReflexSetOptions/slSetData, make sure to initialize when changing one of the other Reflex options during frame limiting.

    //! It is overridden (ignored) by frameLimitUs if set in sl.reflex.json in non-production builds.

    uint32\_t frameLimitUs = 0;

    //! This should only be enabled in specific scenarios with subtle caveats.

    //! Most integrations should leave it unset unless advised otherwise by the Reflex team

    bool useMarkersToOptimize = false;

    //! Specifies the hot-key which should be used instead of custom message for PC latency marker

    //! Possible values: VK\_F13, VK\_F14, VK\_F15

    uint16\_t virtualKey = 0;

    //! ThreadID for PCL Stats messages

    uint32\_t idThread = 0;

    //! IMPORTANT: New members go here or if optional can be chained in a new struct, see sl\_struct.h for details

SL\_STRUCT\_END()

## sl::log (log.cpp)

### ILog

#### 소스코드

struct ILog

{

    virtual void logva(uint32\_t level, ConsoleForeground color, const char\* file, int line, const char\* func, int type, bool isMetaDataUnique, const char\* fmt, ...) = 0;

    virtual void enableConsole(bool flag) = 0;

    virtual LogLevel getLogLevel() const = 0;

    virtual void setLogLevel(LogLevel level) = 0;

    virtual void setLogPath(const wchar\_t\* path) = 0;

    virtual void setLogName(const wchar\_t\* name) = 0;

    virtual void setLogCallback(void\* logMessageCallback) = 0;

    virtual void setLogMessageDelay(float logMessageDelayMS) = 0;

    virtual const wchar\_t\* getLogPath() = 0;

    virtual const wchar\_t\* getLogName() = 0;

    virtual void flush() = 0;

    virtual void shutdown() = 0;

};

### Log : ILog

#### 소스코드

struct Log : ILog

{

    std::hash<std::string> m\_hash;

    std::atomic<bool> m\_console = false;

    std::atomic<bool> m\_pathInvalid = false;

    std::wstring m\_path;

    std::wstring m\_name;

    LogLevel m\_logLevel = LogLevel::eVerbose;

    std::atomic<bool> m\_consoleActive = false;

    FILE\* m\_file = {};

    PFun\_LogMessageCallback\* m\_logMessageCallback = {};

    thread::WorkerThread\* m\_worker{};

## WinTrust.h

### WINTRUST\_FILE\_INFO

#### 소스코드

//////////////////////////////////////////////////////////////////////////////

//

// WINTRUST\_FILE\_INFO Structure

//----------------------------------------------------------------------------

//  Used when calling WinVerifyTrust against an individual file.

//

typedef struct WINTRUST\_FILE\_INFO\_

{

    DWORD           cbStruct;                   // = sizeof(WINTRUST\_FILE\_INFO)

    LPCWSTR         pcwszFilePath;              // required, file name to be verified

    HANDLE          hFile;                      // optional, open handle to pcwszFilePath

    GUID            \*pgKnownSubject;            // optional: fill if the subject type is known.

} WINTRUST\_FILE\_INFO, \*PWINTRUST\_FILE\_INFO;

## UIData.h

### UIData

#### 소스코드

/// <summary>

/// This enum describes the available anti-aliasing modes. These can be toggled from the UI

/// </summary>

enum class AntiAliasingMode {

    NONE,

    TEMPORAL,

    DLSS,

};

/// <summary>

/// This enum describes the Dynamic Resolution mode used in-game

/// </summary>

enum class RenderingResolutionMode {

    FIXED,

    DYNAMIC,

    COUNT

};

struct UIData

{

    // General

    nvrhi::GraphicsAPI                  GraphicsAPI = nvrhi::GraphicsAPI::D3D12;

    bool                                EnableAnimations = true;

    float                               AnimationSpeed = 1.;

    bool                                EnableVsync = false;

    bool                                VisualiseBuffers = false;

    float                               CpuLoad = 0;

    int                                 GpuLoad = 0;

    donut::math::int2                   Resolution = { 0,0 };

    bool                                Resolution\_changed = false;

    bool                                MouseOverUI = false;

    uint32\_t getNViewports() const { return (uint32\_t)BackBufferExtents.size(); }

    sl::Extent getExtent(uint32\_t fullWidth, uint32\_t fullHeight, uint32\_t uV);

private:

    friend class UIRenderer;

    std::vector<sl::Extent>             BackBufferExtents{};

public:

    // SSAO

    bool                                EnableSsao = true;

    donut::render::SsaoParameters       SsaoParams;

    // Tonemapping

    bool                                 EnableToneMapping = true;

    donut::render::ToneMappingParameters ToneMappingParams;

    // Sky

    bool                                EnableProceduralSky = true;

    donut::render::SkyParameters        SkyParams;

    float                               AmbientIntensity = .2f;

    // Antialising (+TAA)

    AntiAliasingMode                                   AAMode = AntiAliasingMode::NONE;

    donut::render::TemporalAntiAliasingJitter          TemporalAntiAliasingJitter = donut::render::TemporalAntiAliasingJitter::MSAA;

    donut::render::TemporalAntiAliasingParameters      TemporalAntiAliasingParams;

    // Bloom

    bool                                EnableBloom = true;

    float                               BloomSigma = 32.f;

    float                               BloomAlpha = 0.05f;

    // Shadows

    bool                                EnableShadows = true;

    float                               CsmExponent = 4.f;

    // DLSS specific parameters

    float                               DLSS\_Sharpness = 0.f;

    bool                                DLSS\_Supported = false;

    sl::DLSSMode                        DLSS\_Mode = sl::DLSSMode::eOff;

    RenderingResolutionMode             DLSS\_Resolution\_Mode = RenderingResolutionMode::FIXED;

    bool                                DLSS\_Dynamic\_Res\_change = true;

    AntiAliasingMode                    DLSS\_Last\_AA = AntiAliasingMode::NONE;

    bool                                DLSS\_DebugShowFullRenderingBuffer = false;

    bool                                DLSS\_lodbias\_useoveride = false;

    float                               DLSS\_lodbias\_overide = 0.f;

    bool                                DLSS\_always\_use\_extents = false;

    sl::DLSSPreset                      DLSS\_presets[static\_cast<int>(sl::DLSSMode::eCount)] = {};

    sl::DLSSPreset                      DLSS\_last\_presets[static\_cast<int>(sl::DLSSMode::eCount)] = {};

    bool UIData::DLSSPresetsChanged()

    {

        for (int i = 0; i < static\_cast<int>(sl::DLSSMode::eCount); i++)

        {

            if (DLSS\_presets[i] != DLSS\_last\_presets[i])

                return true;

        }

        return false;

    };

    bool UIData::DLSSPresetsAnyNonDefault()

    {

        for (int i = 0; i < static\_cast<int>(sl::DLSSMode::eCount); i++)

        {

            if (DLSS\_presets[i] != sl::DLSSPreset::eDefault)

                return true;

        }

        return false;

    };

    void UIData::DLSSPresetsUpdate()

    {

        for (int i = 0; i < static\_cast<int>(sl::DLSSMode::eCount); i++)

            DLSS\_last\_presets[i] = DLSS\_presets[i];

    };

    void UIData::DLSSPresetsReset()

    {

        for (int i = 0; i < static\_cast<int>(sl::DLSSMode::eCount); i++)

            DLSS\_last\_presets[i] = DLSS\_presets[i] = sl::DLSSPreset::eDefault;

    };

    // NIS specific parameters

    bool                                NIS\_Supported = false;

    sl::NISMode                         NIS\_Mode = sl::NISMode::eOff;

    float                               NIS\_Sharpness = 0.5f;

    // DeepDVC specific parameters

    bool                                DeepDVC\_Supported = false;

    sl::DeepDVCMode                     DeepDVC\_Mode = sl::DeepDVCMode::eOff;

    float                               DeepDVC\_Intensity = 0.5f;

    float                               DeepDVC\_SaturationBoost = 0.75f;

    uint64\_t                            DeepDVC\_VRAM = 0;

    // LATENCY specific parameters

    bool                                REFLEX\_Supported = false;

    bool                                REFLEX\_LowLatencyAvailable = false;

    int                                 REFLEX\_Mode = static\_cast<int>(sl::ReflexMode::eOff);

    int                                 REFLEX\_CapedFPS = 0;

    std::string                         REFLEX\_Stats = "";

    // DLFG specific parameters

    bool                                DLSSG\_Supported = false;

    sl::DLSSGMode                       DLSSG\_mode = sl::DLSSGMode::eOff;

    int                                 DLSSG\_numFrames = 2;

    int                                 DLSSG\_numFramesMaxMultiplier = 4;

    float                               DLSSG\_fps = 0;

    size\_t                              DLSSG\_memory = 0;

    std::string                         DLSSG\_status = "";

    bool                                DLSSG\_cleanup\_needed = false;

    // Latewarp

    bool                                Latewarp\_Supported = false;

    int                                 Latewarp\_active = 0;

};

# Class

## 전역선언 (SLWrapper.h)

### SLWrapper

#### 동작

##### DLSS 기능을 위해 Streamline 기능을 적용한 wrapper 함수

##### m\_Device의 경우 IDevice로 설정하였으며, 추후 signleton으로 관리되는 IDevice와 동기화 함

##### SL 기능 별 옵션은 각 Function별로 제공되는 구조체를 활용해 관리

#### 멤버변수

    bool m\_sl\_initialised = false;

    nvrhi::GraphicsAPI m\_api = nvrhi::GraphicsAPI::D3D12;

    nvrhi::IDevice\* m\_Device = nullptr;

#if DONUT\_WITH\_DX11

    LUID m\_d3d11Luid;

#endif

    bool m\_dlss\_available = false;

    sl::DLSSOptions m\_dlss\_consts{};

    bool m\_nis\_available = false;

    sl::NISOptions m\_nis\_consts{};

    bool m\_deepdvc\_available = false;

    sl::DeepDVCOptions m\_deepdvc\_consts{};

    bool m\_dlssg\_available = false;

    bool m\_dlssg\_triggerswapchainRecreation = false;

    bool m\_dlssg\_shoudLoad = false;

    sl::DLSSGOptions m\_dlssg\_consts{};

    sl::DLSSGState m\_dlssg\_settings{};

    bool m\_latewarp\_available = false;

    bool m\_latewarp\_triggerSwapchainRecreation = false;

    bool m\_latewarp\_shouldLoad = false;

    bool m\_reflex\_available = false;

    sl::ReflexOptions m\_reflex\_consts{};

    bool m\_reflex\_driverFlashIndicatorEnable = false;

    bool m\_pcl\_available = false;

    static sl::Resource allocateResourceCallback(const sl::ResourceAllocationDesc\* resDesc, void\* device);

    static void releaseResourceCallback(sl::Resource\* resource, void\* device);

    sl::FrameToken\* m\_currentFrame;

    sl::ViewportHandle m\_viewport = {0};

## Unknwnbase.h

### IID\_IUnknown

#### 소스코드

    EXTERN\_C const IID IID\_IUnknown;

    extern "C++"

    {

        MIDL\_INTERFACE("00000000-0000-0000-C000-000000000046")

        IUnknown

        {

        public:

            BEGIN\_INTERFACE

            virtual HRESULT STDMETHODCALLTYPE QueryInterface(

                /\* [in] \*/ REFIID riid,

                /\* [iid\_is][out] \*/ \_COM\_Outptr\_ void \_\_RPC\_FAR \*\_\_RPC\_FAR \*ppvObject) = 0;

            virtual ULONG STDMETHODCALLTYPE AddRef( void) = 0;

            virtual ULONG STDMETHODCALLTYPE Release( void) = 0;

            template<class Q>

            HRESULT

#ifdef \_M\_CEE\_PURE

            \_\_clrcall

#else

            STDMETHODCALLTYPE

#endif

            QueryInterface(\_COM\_Outptr\_ Q\*\* pp)

            {

                return QueryInterface(\_\_uuidof(Q), (void \*\*)pp);

            }

            END\_INTERFACE

        };

    } // extern C++

## sl::thread

### WorkerThread

#### 멤버변수

class WorkerThread

{

    std::mutex m\_mtx;

    std::condition\_variable m\_cv; // work queue cv

    bool m\_workAdded = false;

    std::condition\_variable m\_cvf; // flushing cv, no need for flag since we use a timeout for it

    std::atomic<bool> m\_quit = false;

    std::atomic<bool> m\_flush = false;

    size\_t m\_jobCount = 0;

    std::thread m\_thread;

    std::list<std::pair<bool, std::function<void(void)>>> m\_work{};

    std::wstring m\_name;

#### 멤버함수

##### 생성자

###### 스레드 명과, 스레드의 우선순위를 관리함

WorkerThread(const wchar\_t\* name, int priority)

{

    m\_name = name;

    m\_thread = std::thread(&WorkerThread::workerFunction, this);

    if (!SetThreadPriority(m\_thread.native\_handle(), priority))

    {

        SL\_LOG\_WARN("Failed to set thread priority to %d for thread '%S'", priority, name);

    }

    SetThreadDescription(m\_thread.native\_handle(), name);

}

##### 소멸자

~WorkerThread()

{

    {

        std::unique\_lock<std::mutex> lock(m\_mtx);

        m\_quit = true; // set to true so that worker thread can exit its loop

        m\_workAdded = true; // set to true so that worker thread exit its wait after the notify call

    }

    m\_cv.notify\_all(); // wake up thread

    m\_thread.join(); // block until thread exits

}

##### workerFunction()

void workerFunction()

{

    while (!m\_quit)

    {

        std::unique\_lock<std::mutex> lock(m\_mtx);

        if (m\_work.empty())

        {

            // Tell threads waiting on flush that we are done

            m\_cvf.notify\_all();

            // Check if there was work added while the work queue was empty. If added, don't wait. Otherwise, keep waiting until notify + work added

            m\_cv.wait(lock, [this] { return m\_workAdded; });

            m\_workAdded = false;

        }

        else

        {

            auto [perpetual, func] = m\_work.front();

            lock.unlock();

            // NOTE: No need to wrap this in the exception handler

            // since all internal workers are already executing within one.

            func();

            lock.lock();

            // Done, remove from the queue

            m\_work.erase(m\_work.begin());

            // Keep perpetual jobs until flush is requested

            if (!perpetual || m\_flush.load())

            {

                m\_jobCount--;

            }

            else

            {

                // Back to the queue to execute again but after other workloads (if any)

                m\_work.push\_back({ perpetual, func });

            }

        }

    }

}

## donut::app::DeviceManager (DeviceManager.h)

### DeviceManager

#### 멤버변수

protected:

    // useful for apps that require 2 frames worth of simulation data before first render

    // apps should extend the DeviceManager classes, and constructor initialized this to true to opt in to the behavior

    bool m\_SkipRenderOnFirstFrame = false;

    bool m\_windowVisible = false;

    bool m\_windowIsInFocus = true;

    DeviceCreationParameters m\_DeviceParams;

    GLFWwindow \*m\_Window = nullptr;

    bool m\_EnableRenderDuringWindowMovement = false;

    // set to true if running on NV GPU

    bool m\_IsNvidia = false;

    std::list<IRenderPass \*> m\_vRenderPasses;

    // timestamp in seconds for the previous frame

    double m\_PreviousFrameTimestamp = 0.0;

    // current DPI scale info (updated when window moves)

    float m\_DPIScaleFactorX = 1.f;

    float m\_DPIScaleFactorY = 1.f;

    bool m\_RequestedVSync = false;

    bool m\_InstanceCreated = false;

    double m\_AverageFrameTime = 0.0;

    double m\_AverageTimeUpdateInterval = 0.5;

    double m\_FrameTimeSum = 0.0;

    int m\_NumberOfAccumulatedFrames = 0;

    uint32\_t m\_FrameIndex = 0;

    std::vector<nvrhi::FramebufferHandle> m\_SwapChainFramebuffers;

        // GetFrameIndex cannot be used inside of these callbacks, hence the additional passing of frameID

        // Refer to AnimateRenderPresent implementation for more details

        struct PipelineCallbacks {

            std::function<void(DeviceManager&, uint32\_t)> beforeFrame = nullptr;

            std::function<void(DeviceManager&, uint32\_t)> beforeAnimate = nullptr;

            std::function<void(DeviceManager&, uint32\_t)> afterAnimate = nullptr;

            std::function<void(DeviceManager&, uint32\_t)> beforeRender = nullptr;

            std::function<void(DeviceManager&, uint32\_t)> afterRender = nullptr;

            std::function<void(DeviceManager&, uint32\_t)> beforePresent = nullptr;

            std::function<void(DeviceManager&, uint32\_t)> afterPresent = nullptr;

        } m\_callbacks;

    private:

        static DeviceManager\* CreateD3D11();

        static DeviceManager\* CreateD3D12();

        static DeviceManager\* CreateVK();

        std::string m\_WindowTitle;

#if DONUT\_WITH\_AFTERMATH

        AftermathCrashDump m\_AftermathCrashDumper;

#endif

    };

##### 예시

###### A screenshot of a computer AI-generated content may be incorrect.

#### 멤버 함수

##### CreateInstanceInternal()

bool DeviceManager\_DX12::CreateInstanceInternal()

{

    if (!m\_DxgiFactory2)

    {

        HRESULT hres = CreateDXGIFactory2(m\_DeviceParams.enableDebugRuntime ? DXGI\_CREATE\_FACTORY\_DEBUG : 0, IID\_PPV\_ARGS(&m\_DxgiFactory2));

        if (hres != S\_OK)

        {

            donut::log::error("ERROR in CreateDXGIFactory2.\n"

                "For more info, get log from debug D3D runtime: (1) Install DX SDK, and enable Debug D3D from DX Control Panel Utility. (2) Install and start DbgView. (3) Try running the program again.\n");

            return false;

        }

    }

    return true;

}

##### bool DeviceManager::CreateInstance(const InstanceParameters& params)

###### 소스코드

bool DeviceManager::CreateInstance(const InstanceParameters& params)

{

    if (m\_InstanceCreated)

        return true;

    static\_cast<InstanceParameters&>(m\_DeviceParams) = params;

    if (!params.headlessDevice)

    {

        if (!glfwInit())

            return false;

    }

#if DONUT\_WITH\_AFTERMATH

    if (params.enableAftermath)

    {

        m\_AftermathCrashDumper.EnableCrashDumpTracking();

    }

#endif

    m\_InstanceCreated = CreateInstanceInternal();

    return m\_InstanceCreated;

}

###### 동작

메모이제이션 활용 초기화 완료 검사

glfwInit 호출을 통해 GLFW API 초기화 진행 (분석을 진행할지는, 우선 살펴보자)

## DeviceManagerOverride\_DX12 (DeviceManagerOverride\_DX12.cpp)

### DeviceManagerOverride\_DX12

#### 멤버 변수

##### TBD

#### 멤버 함수

##### bool DeviceManagerOverride\_DX12::CreateDevice()

###### 동작

m\_DeviceParams.enableDebugRuntime == true 인 경우

D3D12GetDebugInterface() 실행

EnableDebugLayer() 실행 (이때, 효율적으로 RefCountPtr를 활용)

###### 소스코드

bool DeviceManagerOverride\_DX12::CreateDevice()

{

    if (m\_DeviceParams.enableDebugRuntime)

    {

        RefCountPtr<ID3D12Debug> pDebug;

        HRESULT hr = D3D12GetDebugInterface(IID\_PPV\_ARGS(&pDebug));

        if (SUCCEEDED(hr))

            pDebug->EnableDebugLayer();

        else

            donut::log::warning("Cannot enable DX12 debug runtime, ID3D12Debug is not available.");

    }

    if (m\_DeviceParams.enableGPUValidation)

    {

        RefCountPtr<ID3D12Debug3> debugController3;

        HRESULT hr = D3D12GetDebugInterface(IID\_PPV\_ARGS(&debugController3));

        if (SUCCEEDED(hr))

            debugController3->SetEnableGPUBasedValidation(true);

        else

            donut::log::warning("Cannot enable GPU-based validation, ID3D12Debug3 is not available.");

    }

    m\_DxgiAdapter = FindAdapter(m\_DeviceParams.adapterIndex);

    HRESULT hr = D3D12CreateDevice(

        m\_DxgiAdapter,

        m\_DeviceParams.featureLevel,

        IID\_PPV\_ARGS(&m\_Device12));

    if (FAILED(hr))

    {

        donut::log::error("D3D12CreateDevice failed, error code = 0x%08x", hr);

        return false;

    }

    if (m\_DeviceParams.enableDebugRuntime)

    {

        RefCountPtr<ID3D12InfoQueue> pInfoQueue;

        m\_Device12->QueryInterface(&pInfoQueue);

        if (pInfoQueue)

        {

#ifdef \_DEBUG

            pInfoQueue->SetBreakOnSeverity(D3D12\_MESSAGE\_SEVERITY\_CORRUPTION, true);

            pInfoQueue->SetBreakOnSeverity(D3D12\_MESSAGE\_SEVERITY\_ERROR, true);

#endif

            D3D12\_MESSAGE\_ID disableMessageIDs[] = {

                D3D12\_MESSAGE\_ID\_CLEARDEPTHSTENCILVIEW\_MISMATCHINGCLEARVALUE,

                D3D12\_MESSAGE\_ID\_COMMAND\_LIST\_STATIC\_DESCRIPTOR\_RESOURCE\_DIMENSION\_MISMATCH, // descriptor validation doesn't understand acceleration structures

            };

            D3D12\_INFO\_QUEUE\_FILTER filter = {};

            filter.DenyList.pIDList = disableMessageIDs;

            filter.DenyList.NumIDs = sizeof(disableMessageIDs) / sizeof(disableMessageIDs[0]);

            pInfoQueue->AddStorageFilterEntries(&filter);

        }

    }

    SLWrapper::Get().ProxyToNative(m\_Device12, (void\*\*)&m\_Device\_native);

    SLWrapper::Get().SetDevice\_raw(m\_Device\_native);

    D3D12\_COMMAND\_QUEUE\_DESC queueDesc;

    ZeroMemory(&queueDesc, sizeof(queueDesc));

    queueDesc.Flags = D3D12\_COMMAND\_QUEUE\_FLAG\_NONE;

    queueDesc.Type = D3D12\_COMMAND\_LIST\_TYPE\_DIRECT;

    queueDesc.NodeMask = 1;

    hr = m\_Device12->CreateCommandQueue(&queueDesc, IID\_PPV\_ARGS(&m\_GraphicsQueue));

    HR\_RETURN(hr)

    m\_GraphicsQueue->SetName(L"Graphics Queue");

    if (m\_DeviceParams.enableComputeQueue)

    {

        queueDesc.Type = D3D12\_COMMAND\_LIST\_TYPE\_COMPUTE;

        hr = m\_Device12->CreateCommandQueue(&queueDesc, IID\_PPV\_ARGS(&m\_ComputeQueue));

        HR\_RETURN(hr)

        m\_ComputeQueue->SetName(L"Compute Queue");

    }

    if (m\_DeviceParams.enableCopyQueue)

    {

        queueDesc.Type = D3D12\_COMMAND\_LIST\_TYPE\_COPY;

        hr = m\_Device12->CreateCommandQueue(&queueDesc, IID\_PPV\_ARGS(&m\_CopyQueue));

        HR\_RETURN(hr)

        m\_CopyQueue->SetName(L"Copy Queue");

    }

    nvrhi::d3d12::DeviceDesc deviceDesc;

    deviceDesc.errorCB = &DefaultMessageCallback::GetInstance();

    deviceDesc.pDevice = m\_Device12;

    deviceDesc.pGraphicsCommandQueue = m\_GraphicsQueue;

    deviceDesc.pComputeCommandQueue = m\_ComputeQueue;

    deviceDesc.pCopyCommandQueue = m\_CopyQueue;

#if DONUT\_WITH\_AFTERMATH

    deviceDesc.aftermathEnabled = m\_DeviceParams.enableAftermath;

#endif

    m\_NvrhiDevice = nvrhi::d3d12::createDevice(deviceDesc);

    if (m\_DeviceParams.enableNvrhiValidationLayer)

    {

        m\_NvrhiDevice = nvrhi::validation::createValidationLayer(m\_NvrhiDevice);

    }

    return true;

}

##### static IDXGIAdapter\* FindAdapter(int adapterID)

###### 동작

Hooking이 적용되어 sl.interposer 내 CreateDXGIFactory1 실행 (dxgi.cpp)

###### 소스코드

// Find an adapter whose name contains the given string.

static IDXGIAdapter\* FindAdapter(int adapterID)

{

    IDXGIFactory1\* DXGIFactory;

    HRESULT hres = CreateDXGIFactory1(IID\_PPV\_ARGS(&DXGIFactory));

    if (hres != S\_OK)

    {

        donut::log::error("ERROR in CreateDXGIFactory.\n"

            "For more info, get log from debug D3D runtime: (1) Install DX SDK, and enable Debug D3D from DX Control Panel Utility. (2) Install and start DbgView. (3) Try running the program again.\n");

        return nullptr;

    }

    unsigned int adapterNo = 0;

    while (SUCCEEDED(hres))

    {

        if (adapterID >= 0 && adapterNo != adapterID) continue;

        IDXGIAdapter\* pAdapter;

        hres = DXGIFactory->EnumAdapters(adapterNo, &pAdapter);

        if (SUCCEEDED(hres))

        {

            DXGI\_ADAPTER\_DESC aDesc;

            pAdapter->GetDesc(&aDesc);

            std::wstring aName = aDesc.Description;

            std::wstring aLuid = std::to\_wstring(aDesc.AdapterLuid.HighPart) + std::to\_wstring(aDesc.AdapterLuid.LowPart);

            if (IsNvDeviceID(aDesc.VendorId))

            {

                return pAdapter;

            }

        }

        adapterNo++;

    }

    return nullptr;

}

## donut::engin::animation (KeyframeAnimation.h)

### Keyframe (struct)

#### 소스코드

struct Keyframe

{

    float time = 0.f;

    dm::float4 value = 0.f;

    dm::float4 inTangent = 0.f;

    dm::float4 outTangent = 0.f;

};

### InterpolationMode (enum class)

#### 소스코드

enum class InterpolationMode

{

    Step,

    Linear,

    Slerp,

    CatmullRomSpline,

    HermiteSpline

};

### Sampler

#### 소스코드

class Sampler

{

protected:

    std::vector<Keyframe> m\_Keyframes;

    InterpolationMode m\_Mode = InterpolationMode::Step;

public:

    Sampler() = default;

    virtual ~Sampler() = default;

    std::optional<dm::float4> Evaluate(float time, bool extrapolateLastValues = false) const;

    [[nodiscard]] std::vector<Keyframe>& GetKeyframes() { return m\_Keyframes; }

    void AddKeyframe(const Keyframe keyframe);

    [[nodiscard]] InterpolationMode GetMode() const { return m\_Mode; }

    void SetInterpolationMode(InterpolationMode mode) { m\_Mode = mode; }

    [[nodiscard]] float GetStartTime() const;

    [[nodiscard]] float GetEndTime() const;

    void Load(Json::Value& node);

};

### Sequence

#### 소스코드

class Sequence

{

protected:

    std::unordered\_map<std::string, std::shared\_ptr<Sampler>> m\_Tracks;

    float m\_Duration = 0.f;

public:

    Sequence() = default;

    virtual ~Sequence() = default;

    std::shared\_ptr<Sampler> GetTrack(const std::string& name)

    {

        return m\_Tracks[name];

    }

    std::optional<dm::float4> Evaluate(const std::string& name, float time, bool extrapolateLastValues = false);

    void AddTrack(const std::string& name, const std::shared\_ptr<Sampler>& track);

    [[nodiscard]] float GetDuration() const { return m\_Duration; }

    void Load(Json::Value& node);

};

## donut::engin::animation (SceneGraph.h)

### DirtyFlags

### SceneContentFlags

### SceneGraph – 메쉬 인스턴스, 스킨드 메쉬 인스턴스, 애니메이션, 카메라 객체, 조명 관리

#### 소스코드

class SceneGraph : public std::enable\_shared\_from\_this<SceneGraph>

{

private:

    friend class SceneGraphNode;

    std::shared\_ptr<SceneGraphNode> m\_Root;

    ResourceTracker<Material> m\_Materials;

    ResourceTracker<MeshInfo> m\_Meshes;

    size\_t m\_GeometryCount = 0;

    size\_t m\_GeometryInstancesCount = 0;

    std::vector<std::shared\_ptr<MeshInstance>> m\_MeshInstances;

    std::vector<std::shared\_ptr<SkinnedMeshInstance>> m\_SkinnedMeshInstances;

    std::vector<std::shared\_ptr<SceneGraphAnimation>> m\_Animations;

    std::vector<std::shared\_ptr<SceneCamera>> m\_Cameras;

    std::vector<std::shared\_ptr<Light>> m\_Lights;

protected:

    virtual void RegisterLeaf(const std::shared\_ptr<SceneGraphLeaf>& leaf);

    virtual void UnregisterLeaf(const std::shared\_ptr<SceneGraphLeaf>& leaf);

public:

    SceneGraph() = default;

    virtual ~SceneGraph() = default;

    SceneResourceCallback<MeshInfo> OnMeshAdded;

    SceneResourceCallback<MeshInfo> OnMeshRemoved;

    SceneResourceCallback<Material> OnMaterialAdded;

    SceneResourceCallback<Material> OnMaterialRemoved;

    [[nodiscard]] const std::shared\_ptr<SceneGraphNode>& GetRootNode() const { return m\_Root; }

    [[nodiscard]] const ResourceTracker<Material>& GetMaterials() const { return m\_Materials; }

    [[nodiscard]] const ResourceTracker<MeshInfo>& GetMeshes() const { return m\_Meshes; }

    [[nodiscard]] const size\_t GetGeometryCount() const { return m\_GeometryCount; }

    [[nodiscard]] const size\_t GetGeometryInstancesCount() const { return m\_GeometryInstancesCount; }

    [[nodiscard]] const std::vector<std::shared\_ptr<MeshInstance>>& GetMeshInstances() const { return m\_MeshInstances; }

    [[nodiscard]] const std::vector<std::shared\_ptr<SkinnedMeshInstance>>& GetSkinnedMeshInstances() const { return m\_SkinnedMeshInstances; }

    [[nodiscard]] const std::vector<std::shared\_ptr<SceneGraphAnimation>>& GetAnimations() const { return m\_Animations; }

    [[nodiscard]] const std::vector<std::shared\_ptr<SceneCamera>>& GetCameras() const { return m\_Cameras; }

    [[nodiscard]] const std::vector<std::shared\_ptr<Light>>& GetLights() const { return m\_Lights; }

    [[nodiscard]] bool HasPendingStructureChanges() const { return m\_Root && (m\_Root->m\_Dirty & SceneGraphNode::DirtyFlags::SubgraphStructure) != 0; }

    [[nodiscard]] bool HasPendingTransformChanges() const { return m\_Root && (m\_Root->m\_Dirty & (SceneGraphNode::DirtyFlags::SubgraphTransforms | SceneGraphNode::DirtyFlags::SubgraphPrevTransforms)) != 0; }

    // Replaces the current root node of the graph with the new one.

    std::shared\_ptr<SceneGraphNode> SetRootNode(const std::shared\_ptr<SceneGraphNode>& root);

    // Attaches a node and its subgraph to the parent.

    // If the node is already attached to this or other graph, a deep copy of the subgraph is made first.

    std::shared\_ptr<SceneGraphNode> Attach(const std::shared\_ptr<SceneGraphNode>& parent, const std::shared\_ptr<SceneGraphNode>& child);

    // Creates a node holding the provided leaf and attaches it to the parent.

    std::shared\_ptr<SceneGraphNode> AttachLeafNode(const std::shared\_ptr<SceneGraphNode>& parent, const std::shared\_ptr<SceneGraphLeaf>& leaf);

    // Removes the node and its subgraph from the graph.

    // When preserveOrder is 'false', the order of node's siblings may be changed during this operation to improve performance.

    std::shared\_ptr<SceneGraphNode> Detach(const std::shared\_ptr<SceneGraphNode>& node, bool preserveOrder = false);

    // Finds a node whose path (sequence of nested node names) matches the provided path,

    // relative to the 'context' node or the root if 'context' is NULL.

    // If the path starts with / the search starts at the root, and the 'context' parameter is ignored.

    // Parent references with .. are supported.

    // If multiple nodes within one parent have the same name matching that component of the path, only the first node will be considered.

    [[nodiscard]] std::shared\_ptr<SceneGraphNode> FindNode(const std::filesystem::path& path, SceneGraphNode\* context = nullptr) const;

    void Refresh(uint32\_t frameIndex);

};

### SceneGraphNode - 노드에 대한 회전, 위치, 바운딩 박스, flag 정보들을 관리

#### 소스코드

class SceneGraphNode final : public std::enable\_shared\_from\_this<SceneGraphNode>

{

public:

    enum struct DirtyFlags : uint32\_t

    {

        None                    = 0,

        LocalTransform          = 0x01,

        PrevTransform           = 0x02,

        Leaf                    = 0x04,

        SubgraphStructure       = 0x08,

        SubgraphTransforms      = 0x10,

        SubgraphPrevTransforms  = 0x20,

        SubgraphContentUpdate   = 0x40,

        SubgraphMask            = (SubgraphStructure | SubgraphTransforms | SubgraphPrevTransforms | SubgraphContentUpdate)

    };

private:

    friend class SceneGraph;

    std::weak\_ptr<SceneGraph> m\_Graph;

    SceneGraphNode\* m\_Parent = nullptr;

    std::vector<std::shared\_ptr<SceneGraphNode>> m\_Children;

    std::shared\_ptr<SceneGraphLeaf> m\_Leaf;

    std::string m\_Name;

    dm::daffine3 m\_LocalTransform = dm::daffine3::identity();

    dm::daffine3 m\_GlobalTransform = dm::daffine3::identity();

    dm::affine3 m\_GlobalTransformFloat = dm::affine3::identity();

    dm::daffine3 m\_PrevLocalTransform = dm::daffine3::identity();

    dm::daffine3 m\_PrevGlobalTransform = dm::daffine3::identity();

    dm::affine3 m\_PrevGlobalTransformFloat = dm::affine3::identity();

    dm::dquat m\_Rotation = dm::dquat::identity();

    dm::double3 m\_Scaling = 1.0;

    dm::double3 m\_Translation = 0.0;

    dm::box3 m\_GlobalBoundingBox = dm::box3::empty();

    bool m\_HasLocalTransform = false;

    DirtyFlags m\_Dirty = DirtyFlags::None;

    SceneContentFlags m\_LeafContent = SceneContentFlags::None;

    SceneContentFlags m\_SubgraphContent = SceneContentFlags::None;

    void UpdateLocalTransform();

    void PropagateDirtyFlags(SceneGraphNode::DirtyFlags flags);

public:

    SceneGraphNode() = default;

    /\* non-virtual \*/ ~SceneGraphNode() = default;

    [[nodiscard]] const dm::dquat& GetRotation() const { return m\_Rotation; }

    [[nodiscard]] const dm::double3& GetScaling() const { return m\_Scaling; }

    [[nodiscard]] const dm::double3& GetTranslation() const { return m\_Translation; }

    [[nodiscard]] const dm::daffine3& GetLocalToParentTransform() const { return m\_LocalTransform; }

    [[nodiscard]] const dm::daffine3& GetLocalToWorldTransform() const { return m\_GlobalTransform; }

    [[nodiscard]] const dm::affine3& GetLocalToWorldTransformFloat() const { return m\_GlobalTransformFloat; }

    [[nodiscard]] const dm::daffine3& GetPrevLocalToParentTransform() const { return m\_PrevLocalTransform; }

    [[nodiscard]] const dm::daffine3& GetPrevLocalToWorldTransform() const { return m\_PrevGlobalTransform; }

    [[nodiscard]] const dm::affine3& GetPrevLocalToWorldTransformFloat() const { return m\_PrevGlobalTransformFloat; }

    [[nodiscard]] const dm::box3& GetGlobalBoundingBox() const { return m\_GlobalBoundingBox; }

    [[nodiscard]] DirtyFlags GetDirtyFlags() const { return m\_Dirty; }

    [[nodiscard]] SceneContentFlags GetLeafContentFlags() const { return m\_LeafContent; }

    [[nodiscard]] SceneContentFlags GetSubgraphContentFlags() const { return m\_SubgraphContent; }

    [[nodiscard]] SceneGraphNode\* GetParent() const { return m\_Parent; }

    [[nodiscard]] SceneGraphNode\* GetChild(size\_t index) const { return (index < m\_Children.size()) ? m\_Children[index].get() : nullptr; }

    [[nodiscard]] size\_t GetNumChildren() const { return m\_Children.size(); }

    [[nodiscard]] const std::shared\_ptr<SceneGraphLeaf>& GetLeaf() const { return m\_Leaf; }

    [[nodiscard]] const std::string& GetName() const { return m\_Name; }

    [[nodiscard]] std::shared\_ptr<SceneGraph> GetGraph() const { return m\_Graph.lock(); }

    [[nodiscard]] std::filesystem::path GetPath() const;

    void InvalidateContent();

    void SetTransform(const dm::double3\* translation, const dm::dquat\* rotation, const dm::double3\* scaling);

    void SetScaling(const dm::double3& scaling);

    void SetRotation(const dm::dquat& rotation);

    void SetTranslation(const dm::double3& translation);

    void SetLeaf(const std::shared\_ptr<SceneGraphLeaf>& leaf);

    void SetName(const std::string& name);

    // Non-copyable and non-movable

    SceneGraphNode(const SceneGraphNode&) = delete;

    SceneGraphNode(const SceneGraphNode&&) = delete;

    SceneGraphNode& operator=(const SceneGraphNode&) = delete;

    SceneGraphNode& operator=(const SceneGraphNode&&) = delete;

};

### SceneGraphLeaf

#### 소스코드

class SceneGraphLeaf

{

private:

    friend class SceneGraphNode;

    std::weak\_ptr<SceneGraphNode> m\_Node;

protected:

    SceneGraphLeaf() = default;

public:

    virtual ~SceneGraphLeaf() = default;

    [[nodiscard]] SceneGraphNode\* GetNode() const { return m\_Node.lock().get(); }

    [[nodiscard]] std::shared\_ptr<SceneGraphNode> GetNodeSharedPtr() const { return m\_Node.lock(); }

    [[nodiscard]] virtual dm::box3 GetLocalBoundingBox() { return dm::box3::empty(); }

    [[nodiscard]] virtual std::shared\_ptr<SceneGraphLeaf> Clone() = 0;

    [[nodiscard]] virtual SceneContentFlags GetContentFlags() const { return SceneContentFlags::None; }

    [[nodiscard]] const std::string& GetName() const;

    void SetName(const std::string& name) const;

    virtual void Load(const Json::Value& node) { }

    virtual bool SetProperty(const std::string& name, const dm::float4& value) { return false; }

    // Non-copyable and non-movable

    SceneGraphLeaf(const SceneGraphLeaf&) = delete;

    SceneGraphLeaf(const SceneGraphLeaf&&) = delete;

    SceneGraphLeaf& operator=(const SceneGraphLeaf&) = delete;

    SceneGraphLeaf& operator=(const SceneGraphLeaf&&) = delete;

};

### SceneGraphAnimationChannel

#### 소스코드

class SceneGraphAnimationChannel

{

private:

    std::shared\_ptr<animation::Sampler> m\_Sampler;

    std::weak\_ptr<SceneGraphNode> m\_TargetNode;

    std::weak\_ptr<Material> m\_TargetMaterial;

    AnimationAttribute m\_Attribute;

    std::string m\_LeafPropertyName;

public:

    SceneGraphAnimationChannel(std::shared\_ptr<animation::Sampler> sampler, const std::shared\_ptr<SceneGraphNode>& targetNode, AnimationAttribute attribute)

        : m\_Sampler(std::move(sampler))

        , m\_TargetNode(targetNode)

        , m\_Attribute(attribute)

    { }

    SceneGraphAnimationChannel(std::shared\_ptr<animation::Sampler> sampler, const std::shared\_ptr<Material>& targetMaterial)

        : m\_Sampler(std::move(sampler))

        , m\_TargetMaterial(targetMaterial)

        , m\_Attribute(AnimationAttribute::LeafProperty)

    { }

    [[nodiscard]] bool IsValid() const;

    [[nodiscard]] const std::shared\_ptr<animation::Sampler>& GetSampler() const { return m\_Sampler; }

    [[nodiscard]] AnimationAttribute GetAttribute() const { return m\_Attribute; }

    [[nodiscard]] std::shared\_ptr<SceneGraphNode> GetTargetNode() const { return m\_TargetNode.lock(); }

    [[nodiscard]] const std::string& GetLeafPropertyName() const { return m\_LeafPropertyName; }

    void SetTargetNode(const std::shared\_ptr<SceneGraphNode>& node) { m\_TargetNode = node; }

    void SetLeafProperyName(const std::string& name) { m\_LeafPropertyName = name; }

    bool Apply(float time) const;  // NOLINT(modernize-use-nodiscard)

};

### SceneGraphAnimation

## donut::engine (SceneTypes.h)

### struct Material – 머테리얼 속성 관리

#### 소스코드

struct Material

{

    std::string name;

    std::string modelFileName;      // where this material originated from, e.g. GLTF file name

    int materialIndexInModel = -1;  // index of the material in the model file

    MaterialDomain domain = MaterialDomain::Opaque;

    std::shared\_ptr<LoadedTexture> baseOrDiffuseTexture; // metal-rough: base color; spec-gloss: diffuse color; .a = opacity (both modes)

    std::shared\_ptr<LoadedTexture> metalRoughOrSpecularTexture; // metal-rough: ORM map; spec-gloss: specular color, .a = glossiness

    std::shared\_ptr<LoadedTexture> normalTexture;

    std::shared\_ptr<LoadedTexture> emissiveTexture;

    std::shared\_ptr<LoadedTexture> occlusionTexture;

    std::shared\_ptr<LoadedTexture> transmissionTexture; // see KHR\_materials\_transmission; undefined on specular-gloss materials

    std::shared\_ptr<LoadedTexture> opacityTexture; // for renderers that store opacity or alpha mask separately, overrides baseOrDiffuse.a

    nvrhi::BufferHandle materialConstants;

    dm::float3 baseOrDiffuseColor = 1.f; // metal-rough: base color, spec-gloss: diffuse color (if no texture present)

    dm::float3 specularColor = 0.f; // spec-gloss: specular color

    dm::float3 emissiveColor = 0.f;

    float emissiveIntensity = 1.f; // additional multiplier for emissiveColor

    float metalness = 0.f; // metal-rough only

    float roughness = 0.f; // both metal-rough and spec-gloss

    float opacity = 1.f; // for transparent materials; multiplied by diffuse.a if present

    float alphaCutoff = 0.5f; // for alpha tested materials

    float transmissionFactor = 0.f; // see KHR\_materials\_transmission; undefined on specular-gloss materials

    float normalTextureScale = 1.f;

    float occlusionStrength = 1.f;

    // Toggle between two PBR models: metal-rough and specular-gloss.

    // See the comments on the other fields here.

    bool useSpecularGlossModel = false;

    // Toggles for the textures. Only effective if the corresponding texture is non-null.

    bool enableBaseOrDiffuseTexture = true;

    bool enableMetalRoughOrSpecularTexture = true;

    bool enableNormalTexture = true;

    bool enableEmissiveTexture = true;

    bool enableOcclusionTexture = true;

    bool enableTransmissionTexture = true;

    bool enableOpacityTexture = true;

    bool doubleSided = false;

    // Useful when metalness and roughness are packed into a 2-channel texture for BC5 encoding.

    bool metalnessInRedChannel = false;

    int materialID = 0;

    bool dirty = true; // set this to true to make Scene update the material data

    virtual ~Material() = default;

    void FillConstantBuffer(struct MaterialConstants& constants) const;

    bool SetProperty(const std::string& name, const dm::float4& value);

};

### enum class MaterialDomain : uint8\_t

#### 소스코드

enum class MaterialDomain : uint8\_t

{

    Opaque,

    AlphaTested,

    AlphaBlended,

    Transmissive,

    TransmissiveAlphaTested,

    TransmissiveAlphaBlended,

    Count

};

### struct LoadedTexture

#### 소스코드

struct LoadedTexture

{

    nvrhi::TextureHandle texture;

    TextureAlphaMode alphaMode = TextureAlphaMode::UNKNOWN;

    uint32\_t originalBitsPerPixel = 0;

    DescriptorHandle bindlessDescriptor;

    std::string path;

    std::string mimeType;

};

### enum class TextureAlphaMode

#### 소스코드

enum class TextureAlphaMode

{

    UNKNOWN = 0,

    STRAIGHT = 1,

    PREMULTIPLIED = 2,

    OPAQUE\_ = 3,

    CUSTOM = 4,

};

# Class 별 함수

## donut::app (ApplicationBase.cpp)

### nvrhi::GrapicsAPI donut::app::GetGraphicsAPIFromCommandLine(int argc, const char\* const \* argv)

#### 핵심 동작:

##### 환경을 분석하여 적절한 API 설정을 반환함

#### 동작:

##### 콘솔을 통해 전달된 매개변수의 API를 선택 후 반환

##### 선언된 매개변수가 존재하지 않는 경우, Define으로 설정된 API를 선택하여 반환

##### Define으로 정의한 매개변수가 없는 경우, Error 출력

#### - 소스 코드

nvrhi::GraphicsAPI donut::app::GetGraphicsAPIFromCommandLine(int argc, const char\* const\* argv)

{

    for (int n = 1; n < argc; n++)

    {

        const char\* arg = argv[n];

        if (!strcmp(arg, "-d3d11") || !strcmp(arg, "-dx11") || !strcmp(arg, "--d3d11") || !strcmp(arg, "--dx11"))

            return nvrhi::GraphicsAPI::D3D11;

        else if (!strcmp(arg, "-d3d12") || !strcmp(arg, "-dx12") || !strcmp(arg, "--d3d12") || !strcmp(arg, "--dx12"))

            return nvrhi::GraphicsAPI::D3D12;

        else if(!strcmp(arg, "-vk") || !strcmp(arg, "-vulkan") || !strcmp(arg, "--vk") || !strcmp(arg, "--vulkan"))

            return nvrhi::GraphicsAPI::VULKAN;

    }

#if DONUT\_WITH\_DX12

    return nvrhi::GraphicsAPI::D3D12;

#elif DONUT\_WITH\_VULKAN

    return nvrhi::GraphicsAPI::VULKAN;

#elif DONUT\_WITH\_DX11

    return nvrhi::GraphicsAPI::D3D11;

#else

    #error "No Graphics API defined"

#endif

}

## main

### bool ProcessCommandLine(int argc, const char\* const\* argv, donut::app::DeviceCreationParameters& deviceParams, std::string& sceneName, bool& checkSig, bool& enableSLlog)

#### 핵심 동작: 입력된 매개변수를 활용해 deviceParams를 설정

#### 동작:

##### -width 입력 시: backBufferWidth = std::stoi(argv[++i])

##### -height 입력 시: backBufferHeight = std::stoi(argv[++i])

##### -fullscreen 입력 시: startFullscreen = true

##### -debug 입력 시:

###### enableDebugRuntime = true

###### enableNvrhiValidationLayer = true

##### -verbose 입력 시: donut::log::SetMinSeverity(donut::log::Severity::Info) 함수 호출

##### -logToFile 입력 시:

###### log\_file = std::ofstream(“log.txt”)

###### g\_Callback = &logToFile

##### -noSigCheck 입력 시: checkSig = false

##### -vsync 입력 시: vsyncEnable = true

##### -sllog 입력 시: enableSLlog = true

##### -scene 입력 시: sceneName = argv[i] (버그 아닌가?)

##### -adapter 입력 시: adapterIndex = std::stoi(std::string(argv[++i])) (int형을 제작하는 과정에서 string을 만드는 과정을 거치는 방식)

##### 옵션 미 일치 시: donut::log::warning(“Unrecognized option: “, argv[i]) 호출

bool ProcessCommandLine(int argc, const char\* const\* argv, donut::app::DeviceCreationParameters& deviceParams, std::string& sceneName, bool& checkSig, bool& enableSLlog)

{

    for (int i = 1; i < argc; i++)

    {

        if (!\_stricmp(argv[i], "-width"))

        {

            deviceParams.backBufferWidth = std::stoi(argv[++i]);

        }

        else if (!\_stricmp(argv[i], "-height"))

        {

            deviceParams.backBufferHeight = std::stoi(argv[++i]);

        }

        else if (!\_stricmp(argv[i], "-fullscreen"))

        {

            deviceParams.startFullscreen = true;

        }

        else if (!\_stricmp(argv[i], "-debug"))

        {

            deviceParams.enableDebugRuntime = true;

            deviceParams.enableNvrhiValidationLayer = true;

        }

        else if (!\_stricmp(argv[i], "-verbose"))

        {

            donut::log::SetMinSeverity(donut::log::Severity::Info);

        }

        else if (!\_stricmp(argv[i], "-logToFile"))

        {

            log\_file = std::ofstream("log.txt");

            donut::log::SetCallback(&logToFile);

        }

        else if (!\_stricmp(argv[i], "-noSigCheck"))

        {

            checkSig = false;

        }

        else if (!\_stricmp(argv[i], "-vsync"))

        {

            deviceParams.vsyncEnabled = true;

        }

        else if (!\_stricmp(argv[i], "-sllog"))

        {

            enableSLlog = true;

        }

        else if (!\_stricmp(argv[i], "-scene"))

        {

            sceneName = argv[i];

        }

        else if (!\_stricmp(argv[i], "-adapter"))

        {

            auto temp = std::string(argv[++i]);

            deviceParams.adapterIndex = std::stoi(temp);

        }

        else

        {

            donut::log::warning("Unrecognized option: ", argv[i]);

        }

    }

    return true;

}

### void logToFile(donut::log::Severity s, char const\* txt)

#### 동작

##### log\_file이 열린 상태 확인

###### 입력된 txt를 string으로 치환

###### log\_file 경로에 문자열 출력

void logToFile(donut::log::Severity s, char const\* txt) {

    if (log\_file.is\_open()) {

        auto str = std::string(txt);

        log\_file << str.substr(0, str.size() - 1).c\_str() << std::endl;

    }

};

## donut::log

### void DefaultCallback(Severity severity, const char\* message)

#### 동작

##### serverity에 따른 Prefix 설정

##### g\_MessageBufferSize(default: 4096)에 따른 메모리 공간 선언 (이때, 4096을 할당한 이유는 페이징과 메모리 파편화를 고려하여 할당한 것으로 판단됨)

##### serverity와 message를 하나로 concat

##### g\_LogMutex를 할용해 lock\_guard 적용

##### g\_OutputToDebug == true 시, 디버그 텍스트 출력

##### g\_OutputToMessageBox == true 시, 메시지박스 텍스트 출력

##### g\_OutputToConsole == ture 시, fprintf 출력

##### serverity == Severity::Faatal 시, abort() 호출

#### 소스코드

    void DefaultCallback(Severity severity, const char\* message)

    {

        const char\* severityText = "";

        switch (severity)

        {

        case Severity::Debug: severityText = "DEBUG";  break;

        case Severity::Info: severityText = "INFO";  break;

        case Severity::Warning: severityText = "WARNING"; break;

        case Severity::Error: severityText = "ERROR"; break;

        case Severity::Fatal: severityText = "FATAL ERROR"; break;

default:

break;

        }

        char buf[g\_MessageBufferSize];

        snprintf(buf, std::size(buf), "%s: %s", severityText, message);

        {

            std::lock\_guard<std::mutex> lockGuard(g\_LogMutex);

#if \_WIN32

            if (g\_OutputToDebug)

            {

                OutputDebugStringA(buf);

                OutputDebugStringA("\n");

            }

            if (g\_OutputToMessageBox)

            {

                if (severity == Severity::Error || severity == Severity::Fatal)

                {

                    MessageBoxA(0, buf, g\_ErrorMessageCaption.c\_str(), MB\_ICONERROR);

                }

            }

#endif

            if (g\_OutputToConsole)

            {

                if (severity == Severity::Error || severity == Severity::Fatal)

                    fprintf(stderr, "%s\n", buf);

                else

                    fprintf(stdout, "%s\n", buf);

            }

        }

        if (severity == Severity::Fatal)

            abort();

    }

### void debug/info/waring/error/fatal(const char\* fmt...)

#### 동작

##### g\_MinSeverity와 Serverity 검사

###### g\_MinSeverity보다 작은 경우 동작 종료

##### g\_MessageBufferSize(defualt: 4096) 사이즈의 char 배열 생성

###### 참고사항: 4096은 메모리 파편화 시 효율성 개선을 위함

##### 입력된 데이터를 가공한 후 g\_Callback함수로 데이터 전달

#### 소스코드

void info(const char\* fmt...)

{

    if (static\_cast<int>(g\_MinSeverity) > static\_cast<int>(Severity::Info))

        return;

    char buffer[g\_MessageBufferSize];

    va\_list args;

    va\_start(args, fmt);

    vsnprintf(buffer, std::size(buffer), fmt, args);

    g\_Callback(Severity::Info, buffer);

    va\_end(args);

}

### void message

#### 소스코드

void message(Severity severity, const char\* fmt...)

{

    if (static\_cast<int>(g\_MinSeverity) > static\_cast<int>(severity))

        return;

    char buffer[g\_MessageBufferSize];

    va\_list args;

    va\_start(args, fmt);

    vsnprintf(buffer, std::size(buffer), fmt, args);

    g\_Callback(severity, buffer);

    va\_end(args);

}

## SLWrapper

### void releaseResourceCallback(sl::Resource\* resource, void\* device)

#### 소스코드

void SLWrapper::releaseResourceCallback(sl::Resource\* resource, void\* device)

{

    if (resource)

    {

        auto i = (IUnknown\*)resource->native;

        i->Release();

    }

};

### void allocateResourceCallback(const sl::ResourceAllocationDesc\* resDesc, void\* device)

#### 소스코드

sl::Resource SLWrapper::allocateResourceCallback(const sl::ResourceAllocationDesc\* resDesc, void\* device) {

    sl::Resource res = {};

    if (device == nullptr) {

        log::warning("No device available for allocation.");

        return res;

    }

    bool isBuffer = (resDesc->type == sl::ResourceType::eBuffer);

    if (isBuffer) {

#if DONUT\_WITH\_DX11

        if (Get().m\_api == nvrhi::GraphicsAPI::D3D11)

        {

            D3D11\_BUFFER\_DESC\* desc = (D3D11\_BUFFER\_DESC\*)resDesc->desc;

            ID3D11Device\* pd3d11Device = (ID3D11Device\*)device;

            ID3D11Buffer\* pbuffer;

            bool success = SUCCEEDED(pd3d11Device->CreateBuffer(desc, nullptr, &pbuffer));

            if (!success) log::error("Failed to create buffer in SL allocation callback");

            res.type = resDesc->type;

            res.native = pbuffer;

        }

#endif

#if DONUT\_WITH\_DX12

        if (Get().m\_api == nvrhi::GraphicsAPI::D3D12)

        {

            D3D12\_RESOURCE\_DESC\* desc = (D3D12\_RESOURCE\_DESC\*)resDesc->desc;

            D3D12\_HEAP\_PROPERTIES\* heap = (D3D12\_HEAP\_PROPERTIES\*)resDesc->heap;

            D3D12\_RESOURCE\_STATES state = (D3D12\_RESOURCE\_STATES)resDesc->state;

            ID3D12Device\* pd3d12Device = (ID3D12Device\*)device;

            ID3D12Resource\* pbuffer;

            bool success = SUCCEEDED(pd3d12Device->CreateCommittedResource(heap, D3D12\_HEAP\_FLAG\_NONE, desc, state, nullptr, IID\_PPV\_ARGS(&pbuffer)));

            if (!success) log::error("Failed to create buffer in SL allocation callback");

            res.type = resDesc->type;

            res.native = pbuffer;

        }

#endif

    }

    else {

#if DONUT\_WITH\_DX11

        if (Get().m\_api == nvrhi::GraphicsAPI::D3D11)

        {

            D3D11\_TEXTURE2D\_DESC\* desc = (D3D11\_TEXTURE2D\_DESC\*)resDesc->desc;

            ID3D11Device\* pd3d11Device = (ID3D11Device\*)device;

            ID3D11Texture2D\* ptexture;

            bool success = SUCCEEDED(pd3d11Device->CreateTexture2D(desc, nullptr, &ptexture));

            if (!success) log::error("Failed to create texture in SL allocation callback");

            res.type = resDesc->type;

            res.native = ptexture;

        }

#endif

#if DONUT\_WITH\_DX12

        if (Get().m\_api == nvrhi::GraphicsAPI::D3D12)

        {

            D3D12\_RESOURCE\_DESC\* desc = (D3D12\_RESOURCE\_DESC\*)resDesc->desc;

            D3D12\_RESOURCE\_STATES state = (D3D12\_RESOURCE\_STATES)resDesc->state;

            D3D12\_HEAP\_PROPERTIES\* heap = (D3D12\_HEAP\_PROPERTIES\*)resDesc->heap;

            ID3D12Device\* pd3d12Device = (ID3D12Device\*)device;

            ID3D12Resource\* ptexture;

            D3D12\_CLEAR\_VALUE\* pClearValue = nullptr;

            D3D12\_CLEAR\_VALUE clearValue;

            // specify the clear value to avoid D3D warnings on ClearRenderTarget()

            if (desc->Flags & D3D12\_RESOURCE\_FLAG\_ALLOW\_RENDER\_TARGET)

            {

                clearValue.Format = desc->Format;

                memset(clearValue.Color, 0, sizeof(clearValue.Color));

                pClearValue = &clearValue;

            }

            bool success = SUCCEEDED(pd3d12Device->CreateCommittedResource(heap, D3D12\_HEAP\_FLAG\_NONE, desc, state, pClearValue, IID\_PPV\_ARGS(&ptexture)));

            if (!success) log::error("Failed to create texture in SL allocation callback");

            res.type = resDesc->type;

            res.native = ptexture;

        }

#endif

    }

    return res;

}

### void logFunctionCallback(sl::LogType type, const char\* msg)

#### 소스코드

void logFunctionCallback(sl::LogType type, const char\* msg) {

    if (type == sl::LogType::eError) {

        // Add a breakpoint here to break on errors

        donut::log::error(msg);

    }

    if (type == sl::LogType::eWarn) {

        // Add a breakpoint here to break on warnings

        donut::log::warning(msg);

    }

    else {

        donut::log::info(msg);

    }

}

### std::wstring GetSlInterposerDllLocation()

#### 소스코드

std::wstring GetSlInterposerDllLocation() {

    wchar\_t path[PATH\_MAX] = { 0 };

#ifdef \_WIN32

    if (GetModuleFileNameW(nullptr, path, dim(path)) == 0)

        return std::wstring();

#else // \_WIN32

#error Unsupported platform for GetSlInterposerDllLocation!

#endif // \_WIN32

    auto basePath = std::filesystem::path(path).parent\_path();

    auto dllPath = basePath.wstring().append(L"\\sl.interposer.dll");

    return dllPath;

}

## sl (sl\_core\_api.h)

### #define SL\_ENUM\_OPERATORS\_64(T)

#### 소스코드

#define SL\_ENUM\_OPERATORS\_64(T)                                                         \

inline bool operator&(T a, T b)                                                         \

{                                                                                       \

    return ((uint64\_t)a & (uint64\_t)b) != 0;                                            \

}                                                                                       \

                                                                                        \

inline T& operator&=(T& a, T b)                                                         \

{                                                                                       \

    a = (T)((uint64\_t)a & (uint64\_t)b);                                                 \

    return a;                                                                           \

}                                                                                       \

                                                                                        \

inline T operator|(T a, T b)                                                            \

{                                                                                       \

    return (T)((uint64\_t)a | (uint64\_t)b);                                              \

}                                                                                       \

                                                                                        \

inline T& operator |= (T& lhs, T rhs)                                                   \

{                                                                                       \

    lhs = (T)((uint64\_t)lhs | (uint64\_t)rhs);                                           \

    return lhs;                                                                         \

}                                                                                       \

                                                                                        \

inline T operator~(T a)                                                                 \

{                                                                                       \

    return (T)~((uint64\_t)a);                                                           \

}

### SL\_API sl::Result slInit(const sl::Preferences &pref, uint64\_t sdkVersion = sl::kSDKVersion);

#### 소스코드

APIContext s\_ctx;

sl::Result slInit(const Preferences &pref, uint64\_t sdkVersion)

{

    //! IMPORTANT:

    //!

    //! As explained in sl\_struct.h any new elements must be placed at the end

    //! of each structure and version must be increased or new elements can be

    //! added in a new structure which is then chained. This assert ensures

    //! that new element(s) are NOT added in the middle of a structure.

    static\_assert(offsetof(sl::Preferences, renderAPI) == 136, "new elements can only be added at the end of each structure");

    auto init = [&pref, &sdkVersion]()->sl::Result

    {

        // Setup logging first

        auto log = log::getInterface();

        log->enableConsole(pref.showConsole);

        log->setLogLevel(pref.logLevel);

        log->setLogPath(pref.pathToLogsAndData);

        log->setLogCallback((void\*)pref.logMessageCallback);

        log->setLogName(L"sl.log");

        ConfigureLogOverrides(log);

        if (sl::interposer::hasInterface())

        {

            SL\_LOG\_WARN("Seems like some DX/VK APIs were invoked before slInit()!!! This may result in incorrect behaviour.");

        }

        if (sl::interposer::getInterface()->isEnabled())

        {

            // NOTE: Defaults to true but host can override this since

            // some games might not work correctly with a proxy DXGI.

            bool useDXGIProxy = pref.flags & PreferenceFlags::eUseDXGIFactoryProxy;

            sl::interposer::getInterface()->setUseDXGIProxy(useDXGIProxy);

#ifndef SL\_PRODUCTION

            // Allow overrides via 'sl.interposer.json'

            if (!sl::interposer::getInterface()->getConfigPath().empty())

            {

                auto config = sl::interposer::getInterface()->getConfig();

                if (config.waitForDebugger)

                {

                    SL\_LOG\_INFO("Waiting for debugger to attach ...");

#ifdef SL\_WINDOWS

                    while (!IsDebuggerPresent())

                    {

                        std::this\_thread::sleep\_for(std::chrono::milliseconds(100));

                    }

#endif

                }

            }

#endif

            // Check to see if RenderDoc is present and notify the user

#ifdef SL\_WINDOWS

            HMODULE renderDocMod = GetModuleHandleA("renderdoc.dll");

            if (renderDocMod)

            {

                SL\_LOG\_WARN("RenderDoc has been detected.  As RenderDoc disables NVAPI, any plugins which require NVAPI will be disabled.");

            }

#endif

#ifdef SL\_LINUX

            void\* renderDocMod = dlopen("librenderdoc.so", RTLD\_NOW | RTLD\_NOLOAD);

            if (renderDocMod)

            {

                SL\_LOG\_WARN("RenderDoc has been detected.  As RenderDoc disables NVAPI, any plugins which require NVAPI will be disabled.");

            }

#endif

            auto manager = plugin\_manager::getInterface();

            if (manager->isInitialized())

            {

                SL\_LOG\_ERROR( "slInit must be called before any DXGI/D3D12/D3D11/Vulkan APIs are invoked");

                return Result::eErrorInitNotCalled;

            }

            if (SL\_FAILED(result, manager->setHostSDKVersion(sdkVersion)))

            {

                return result;

            }

            manager->setPreferences(pref);

            param::getInterface()->set(param::global::kPFunAllocateResource, pref.allocateCallback);

            param::getInterface()->set(param::global::kPFunReleaseResource, pref.releaseCallback);

            param::getInterface()->set(param::global::kLogInterface, log::getInterface());

            // Enumerate plugins and check if they are supported or not

            return manager->loadPlugins();

        }

        return Result::eOk;

    };

    SL\_EXCEPTION\_HANDLE\_START

    return init();

    SL\_EXCEPTION\_HANDLE\_END\_RETURN(Result::eErrorExceptionHandler)

}

## sl.cpp

### void ConfigureLogOverrides(log::ILog\* log)

void ConfigureLogOverrides(log::ILog\* log)

{

    // The order of precedence for log overrides is:

    // 1) JSON interposer configuration

    // 2) Environment variables

    // 3) Windows registry

    ConfigureLogOverridesFromRegistry(log);

    ConfigureLogOverridesFromEnvironment(log);

    ConfigureLogOverridesFromInterposerConfig(log);

}

### void ConfigureLogOverridesFromRegistry(log::ILog\* log)

void ConfigureLogOverridesFromRegistry(log::ILog\* log)

{

#ifdef SL\_WINDOWS

    constexpr const wchar\_t\* kRegSubKey = L"SOFTWARE\\NVIDIA Corporation\\Global\\Streamline";

    constexpr const wchar\_t\* kEnableConsoleValue = L"EnableConsoleLogging";

    constexpr const wchar\_t\* kLogLevelValue = L"LogLevel";

    constexpr const wchar\_t\* kLogPathValue = L"LogPath";

    constexpr const wchar\_t\* kLogNameValue = L"LogName";

    bool settingsOverridden = false;

    DWORD registryValue;

    if (extra::getRegistryDword(kRegSubKey, kEnableConsoleValue, &registryValue))

    {

        log->enableConsole(registryValue != 0);

        settingsOverridden = true;

    }

    if (extra::getRegistryDword(kRegSubKey, kLogLevelValue, &registryValue))

    {

        log->setLogLevel(ToLogLevel(registryValue));

        settingsOverridden = true;

    }

    WCHAR registryString[MAX\_PATH];

    if (extra::getRegistryString(kRegSubKey, kLogPathValue, registryString, MAX\_PATH))

    {

        log->setLogPath(registryString);

        settingsOverridden = true;

    }

    if (extra::getRegistryString(kRegSubKey, kLogNameValue, registryString, MAX\_PATH))

    {

        log->setLogName(registryString);

        settingsOverridden = true;

    }

    if (settingsOverridden)

    {

        SL\_LOG\_HINT("Overriding logging settings from registry keys");

    }

#endif

}

### void ConfigureLogOverridesFromEnvironment(log::ILog\* log)

void ConfigureLogOverridesFromEnvironment(log::ILog\* log)

{

    constexpr const char\* kEnableConsoleKey = "SL\_ENABLE\_CONSOLE\_LOGGING";

    constexpr const char\* kLogLevelKey = "SL\_LOG\_LEVEL";

    constexpr const char\* kLogPathKey = "SL\_LOG\_PATH";

    constexpr const char\* kLogNameKey = "SL\_LOG\_NAME";

    std::string value;

    bool settingsOverridden = false;

    if (extra::getEnvVar(kEnableConsoleKey, value)) {

        log->enableConsole(std::atoi(value.c\_str()) != 0);

        settingsOverridden = true;

    }

    if (extra::getEnvVar(kLogLevelKey, value)) {

        log->setLogLevel(ToLogLevel(std::atoi(value.c\_str())));

        settingsOverridden = true;

    }

    if (extra::getEnvVar(kLogPathKey, value)) {

        log->setLogPath(extra::toWStr(value).c\_str());

        settingsOverridden = true;

    }

    if (extra::getEnvVar(kLogNameKey, value)) {

        log->setLogName(extra::toWStr(value).c\_str());

        settingsOverridden = true;

    }

    if (settingsOverridden)

    {

        SL\_LOG\_HINT("Overriding logging settings from environment variables");

    }

}

### 

## sl::ota (ota.cpp)

### void exec

#### 소스코드

    void execThreadProc(const std::wstring command)

    {

        std::string output;

#ifdef SL\_WINDOWS

        HANDLE readPipe, writePipe;

        SECURITY\_ATTRIBUTES security;

        STARTUPINFOW        start;

        PROCESS\_INFORMATION processInfo;

        security.nLength = sizeof(SECURITY\_ATTRIBUTES);

        security.bInheritHandle = true;

        security.lpSecurityDescriptor = NULL;

        if (CreatePipe(

            &readPipe,  // address of variable for read handle

            &writePipe, // address of variable for write handle

            &security,  // pointer to security attributes

            0           // number of bytes reserved for pipe

        )) {

            GetStartupInfoW(&start);

            start.hStdOutput = writePipe;

            start.hStdError = writePipe;

            start.dwFlags = STARTF\_USESTDHANDLES | STARTF\_USESHOWWINDOW;

            start.wShowWindow = SW\_HIDE;

            if (CreateProcessW(NULL,     // pointer to name of executable module

                (wchar\_t\*)command.c\_str(),         // pointer to command line string

                &security,               // pointer to process security attributes

                &security,               // pointer to thread security attributes

                TRUE,                    // handle inheritance flag

                NORMAL\_PRIORITY\_CLASS,   // creation flags

                NULL,                    // pointer to new environment block

                NULL,                    // pointer to current directory name

                &start,                  // pointer to STARTUPINFO

                &processInfo             // pointer to PROCESS\_INFORMATION

            )) {

                DWORD bytesRead = 0, count = 0;

                // 4K buffers to fit nicely on a page :)

                const int BUFF\_SIZE = 0x1000;

                char\* buffer = new char[BUFF\_SIZE];

                output = "";

                // Loop until process is complete, buffering out 4K pages of

                // stderr/stdout data to our output string

                do {

                    DWORD dwAvail = 0;

                    if (PeekNamedPipe(readPipe, NULL, 0, NULL, &dwAvail, NULL)) {

                        if (dwAvail) {

                            if (!ReadFile(readPipe, buffer, BUFF\_SIZE - 1, &bytesRead, NULL))

                            {

                                // failed to read

                                SL\_LOG\_ERROR("Failed ReadFile with error 0x%x", GetLastError());

                                break;

                            }

                            buffer[bytesRead] = '\0';

                            output += buffer;

                            count += bytesRead;

                        }

                        else

                        {

                            // no data available in the pipe

                        }

                    }

                    else

                    {

                        // error, the child process might have ended

                    }

                } while (WaitForSingleObject(processInfo.hProcess, 100) == WAIT\_TIMEOUT);

                delete buffer;

                CloseHandle(processInfo.hThread);

                CloseHandle(processInfo.hProcess);

            }

            else

            {

                SL\_LOG\_ERROR("Failed to create process %ls", command.c\_str());

            }

            CloseHandle(writePipe);

            CloseHandle(readPipe);

        }

        else

        {

            SL\_LOG\_ERROR("Failed to create pipe");

        }

#endif

        SL\_LOG\_VERBOSE("execThreadProc: %ls", command.c\_str());

        // Append a '\n' here so that SL uses "unformatted" logs. The output

        // from the NGX updater is formatted already with timestamps, so we

        // want to remove them before adding our own.

        // Safety note: Passing this directly to the `fmt` parameter of `logva`

        // is safe because the '\n' at the end skips formatting. Using "%s\n"

        // would make the logger skip formatting and print "%s" instead of the

        // intended message.

        if (!output.empty())

        {

            output += '\n';

            SL\_LOG\_VERBOSE(output.c\_str());

        }

    }

## sl::api (plugin.h)

### #define SL\_PLUGIN\_DEFINE(N,V1,V2,JSON,UPDATE\_JSON\_CONFIG, PLUGIN\_NAMESPACE, PLUGIN\_CTX)

#### 소스코드

#define SL\_PLUGIN\_DEFINE(N,V1,V2,JSON,UPDATE\_JSON\_CONFIG, PLUGIN\_NAMESPACE, PLUGIN\_CTX)                    \

namespace api                                                                                              \

{                                                                                                          \

    /\* Created on DLL attached and destroyed on DLL detach from process \*/                                 \

    static Context\* s\_ctx{};                                                                               \

    Context \*getContext() { return s\_ctx; }                                                                \

}                                                                                                          \

                                                                                                           \

SL\_PLUGIN\_CONTEXT\_DEFINE(PLUGIN\_NAMESPACE, PLUGIN\_CTX)                                                     \

                                                                                                           \

bool slOnPluginLoad(sl::param::IParameters \*params, const char\* loaderJSON, const char \*\*pluginJSON)       \

{                                                                                                          \

    if(!sl::PLUGIN\_NAMESPACE::s\_init)                                                                      \

    {                                                                                                      \

        sl::api::s\_ctx = new sl::api::Context(N, sl::V1, sl::V2, nullptr, nullptr,                         \

                                    new json, new json, new json);                                         \

        sl::PLUGIN\_NAMESPACE::s\_ctx = new sl::PLUGIN\_NAMESPACE::PLUGIN\_CTX();                              \

        api::s\_ctx->parameters = params;                                                                   \

        if (!plugin::onLoad(api::getContext(), loaderJSON, JSON))                                          \

        {                                                                                                  \

            return false;                                                                                  \

        }                                                                                                  \

        sl::PLUGIN\_NAMESPACE::s\_init = true;                                                               \

        json& config = \*(json\*)api::getContext()->pluginConfig;                                            \

        UPDATE\_JSON\_CONFIG(config);                                                                        \

        api::s\_ctx->pluginConfigStr = config.dump();                                                       \

    }                                                                                                      \

                                                                                                           \

    \*pluginJSON = api::s\_ctx->pluginConfigStr.c\_str();                                                     \

    return true;                                                                                           \

}                                                                                                          \

                                                                                                           \

}  /\* namespace sl \*/                                                                                      \

/\* Always in global namespace \*/                                                                           \

SL\_EXPORT BOOL APIENTRY DllMain(HMODULE hModule, DWORD fdwReason, LPVOID)                                  \

{                                                                                                          \

    switch (fdwReason)                                                                                     \

    {                                                                                                      \

        case DLL\_PROCESS\_ATTACH:                                                                           \

            break;                                                                                         \

        case DLL\_THREAD\_ATTACH:                                                                            \

            break;                                                                                         \

        case DLL\_THREAD\_DETACH:                                                                            \

            break;                                                                                         \

        case DLL\_PROCESS\_DETACH:                                                                           \

            if (!sl::PLUGIN\_NAMESPACE::s\_init) {                                                           \

                break; /\* if slOnPluginLoad() was never called, no cleanup \*/                              \

            }                                                                                              \

            delete sl::api::s\_ctx;                                                                         \

            delete sl::PLUGIN\_NAMESPACE::s\_ctx;                                                            \

            sl::api::s\_ctx = {};                                                                           \

            sl::PLUGIN\_NAMESPACE::s\_ctx = {};                                                              \

            break;                                                                                         \

    }                                                                                                      \

    return TRUE;                                                                                           \

}                                                                                                          \

namespace sl {

#### 활용 예시

##### dllEntry.cpp

###### SL\_PLUGIN\_DEFINE("sl.dlss", Version(VERSION\_MAJOR, VERSION\_MINOR, VERSION\_PATCH), Version(0, 0, 1), JSON.c\_str(), updateEmbeddedJSON, dlss, DLSSContext)

위 define을 활용해 slOnPluginLoad 기능과, DllMain 구문을 제작함

## sl::security (sl\_security.h)

### bool verifyEmbeddedSignature(const wchar\_t\* pathToFile)

#### 메인 동작

#### 소스코드

//! See https://docs.microsoft.com/en-us/windows/win32/seccrypto/example-c-program--verifying-the-signature-of-a-pe-file

//!

//! IMPORTANT: Always pass in the FULL PATH to the file, relative paths are NOT allowed!

bool verifyEmbeddedSignature(const wchar\_t\* pathToFile)

{

    bool valid = true;

    LONG lStatus = {};

    // Initialize the WINTRUST\_FILE\_INFO structure.

    WINTRUST\_FILE\_INFO FileData;

    memset(&FileData, 0, sizeof(FileData));

    FileData.cbStruct = sizeof(WINTRUST\_FILE\_INFO);

    FileData.pcwszFilePath = pathToFile;

    FileData.hFile = NULL;

    FileData.pgKnownSubject = NULL;

    if (!pfnWinVerifyTrust)

    {

        // We only support Win10+ so we can search for module in system32 directly

        auto hModWintrust = LoadLibraryExW(L"wintrust.dll", NULL, LOAD\_LIBRARY\_SEARCH\_SYSTEM32);

        if (!hModWintrust || !GetProc(hModWintrust, "WinVerifyTrust", pfnWinVerifyTrust))

        {

            return false;

        }

    }

    /\*

    WVTPolicyGUID specifies the policy to apply on the file

    WINTRUST\_ACTION\_GENERIC\_VERIFY\_V2 policy checks:

    1) The certificate used to sign the file chains up to a root

    certificate located in the trusted root certificate store. This

    implies that the identity of the publisher has been verified by

    a certification authority.

    2) In cases where user interface is displayed (which this example

    does not do), WinVerifyTrust will check for whether the

    end entity certificate is stored in the trusted publisher store,

    implying that the user trusts content from this publisher.

    3) The end entity certificate has sufficient permission to sign

    code, as indicated by the presence of a code signing EKU or no

    EKU.

    \*/

    GUID WVTPolicyGUID = WINTRUST\_ACTION\_GENERIC\_VERIFY\_V2;

    WINTRUST\_DATA WinTrustData;

    // Initialize the WinVerifyTrust input data structure.

    // Default all fields to 0.

    memset(&WinTrustData, 0, sizeof(WinTrustData));

    WinTrustData.cbStruct = sizeof(WinTrustData);

    // Use default code signing EKU.

    WinTrustData.pPolicyCallbackData = NULL;

    // No data to pass to SIP.

    WinTrustData.pSIPClientData = NULL;

    // Disable WVT UI.

    WinTrustData.dwUIChoice = WTD\_UI\_NONE;

    // No revocation checking.

    WinTrustData.fdwRevocationChecks = WTD\_REVOKE\_NONE;

    // Verify an embedded signature on a file.

    WinTrustData.dwUnionChoice = WTD\_CHOICE\_FILE;

    // Verify action.

    WinTrustData.dwStateAction = WTD\_STATEACTION\_VERIFY;

    // Verification sets this value.

    WinTrustData.hWVTStateData = NULL;

    // Not used.

    WinTrustData.pwszURLReference = NULL;

    // This is not applicable if there is no UI because it changes

    // the UI to accommodate running applications instead of

    // installing applications.

    WinTrustData.dwUIContext = 0;

    // Set pFile.

    WinTrustData.pFile = &FileData;

    // First verify the primary signature (index 0) to determine how many secondary signatures

    // are present. We use WSS\_VERIFY\_SPECIFIC and dwIndex to do this, also setting

    // WSS\_GET\_SECONDARY\_SIG\_COUNT to have the number of secondary signatures returned.

    WINTRUST\_SIGNATURE\_SETTINGS SignatureSettings = {};

    CERT\_STRONG\_SIGN\_PARA StrongSigPolicy = {};

    SignatureSettings.cbStruct = sizeof(WINTRUST\_SIGNATURE\_SETTINGS);

    SignatureSettings.dwFlags = WSS\_GET\_SECONDARY\_SIG\_COUNT | WSS\_VERIFY\_SPECIFIC;

    SignatureSettings.dwIndex = 0;

    WinTrustData.pSignatureSettings = &SignatureSettings;

    StrongSigPolicy.cbSize = sizeof(CERT\_STRONG\_SIGN\_PARA);

    StrongSigPolicy.dwInfoChoice = CERT\_STRONG\_SIGN\_OID\_INFO\_CHOICE;

    StrongSigPolicy.pszOID = (LPSTR)szOID\_CERT\_STRONG\_SIGN\_OS\_CURRENT;

    WinTrustData.pSignatureSettings->pCryptoPolicy = &StrongSigPolicy;

    // WinVerifyTrust verifies signatures as specified by the GUID  and Wintrust\_Data.

    lStatus = pfnWinVerifyTrust(NULL, &WVTPolicyGUID, &WinTrustData);

    // First signature must be validated by the OS

    valid = lStatus == ERROR\_SUCCESS;

    if (!valid)

    {

        printf("File '%S' is NOT correctly signed - Streamline will not load unsecured modules\n", pathToFile);

    }

    else

    {

        // Now there has to be a secondary one

        valid &= WinTrustData.pSignatureSettings->cSecondarySigs == 1;

        if (!valid)

        {

            printf("File '%S' does not have the secondary NVIDIA signature - Streamline will not load unsecured modules\n", pathToFile);

        }

        else

        {

            // The secondary signature must be from NVIDIA

            valid &= isSignedByNVIDIA(pathToFile);

            if (valid)

            {

                printf("File '%S' is signed by NVIDIA and the signature was verified.\n", pathToFile);

            }

            else

            {

                printf("File '%S' is NOT correctly signed - Streamline will not load unsecured modules\n", pathToFile);

            }

        }

    }

    // Any hWVTStateData must be released by a call with close.

    WinTrustData.dwStateAction = WTD\_STATEACTION\_CLOSE;

    lStatus = pfnWinVerifyTrust(NULL, &WVTPolicyGUID, &WinTrustData);

    return valid;

}

### bool isSignedByNVIDIA(const wchar\_t\* pathToFile)

#### 소스코드

bool isSignedByNVIDIA(const wchar\_t\* pathToFile)

{

    bool valid = false;

    // Now let's make sure this is actually signed by NVIDIA

    DWORD dwEncoding, dwContentType, dwFormatType;

    HCERTSTORE hStore = NULL;

    HCRYPTMSG hMsg = NULL;

    PCMSG\_SIGNER\_INFO pSignerInfo = NULL;

    DWORD dwSignerInfo;

    if (!pfnCertOpenStore)

    {

        // We only support Win10+ so we can search for module in system32 directly

        auto hModCrypt32 = LoadLibraryExW(L"crypt32.dll", NULL, LOAD\_LIBRARY\_SEARCH\_SYSTEM32);

        if (!hModCrypt32 ||

            !GetProc(hModCrypt32, "CryptMsgClose", pfnCryptMsgClose) ||

            !GetProc(hModCrypt32, "CertOpenStore", pfnCertOpenStore) ||

            !GetProc(hModCrypt32, "CertCloseStore", pfnCertCloseStore) ||

            !GetProc(hModCrypt32, "CertFreeCertificateContext", pfnCertFreeCertificateContext) ||

            !GetProc(hModCrypt32, "CertFindCertificateInStore", pfnCertFindCertificateInStore) ||

            !GetProc(hModCrypt32, "CryptMsgGetParam", pfnCryptMsgGetParam) ||

            !GetProc(hModCrypt32, "CryptMsgUpdate", pfnCryptMsgUpdate) ||

            !GetProc(hModCrypt32, "CryptMsgOpenToDecode", pfnCryptMsgOpenToDecode) ||

            !GetProc(hModCrypt32, "CryptQueryObject", pfnCryptQueryObject) ||

            !GetProc(hModCrypt32, "CryptDecodeObjectEx", pfnCryptDecodeObjectEx))

        {

            return false;

        }

    }

    // Get message handle and store handle from the signed file.

    auto bResult = pfnCryptQueryObject(CERT\_QUERY\_OBJECT\_FILE,

        pathToFile,

        CERT\_QUERY\_CONTENT\_FLAG\_PKCS7\_SIGNED\_EMBED,

        CERT\_QUERY\_FORMAT\_FLAG\_BINARY,

        0,

        &dwEncoding,

        &dwContentType,

        &dwFormatType,

        &hStore,

        &hMsg,

        NULL);

    if (!bResult)

    {

        return false;

    }

    // Get signer information size.

    bResult = pfnCryptMsgGetParam(hMsg,

        CMSG\_SIGNER\_INFO\_PARAM,

        0,

        NULL,

        &dwSignerInfo);

    if (!bResult)

    {

        return false;

    }

    // Allocate memory for signer information.

    pSignerInfo = (PCMSG\_SIGNER\_INFO)LocalAlloc(LPTR, dwSignerInfo);

    if (!pSignerInfo)

    {

        return false;

    }

    // Get Signer Information.

    bResult = pfnCryptMsgGetParam(hMsg,

        CMSG\_SIGNER\_INFO\_PARAM,

        0,

        (PVOID)pSignerInfo,

        &dwSignerInfo);

    if (!bResult)

    {

        LocalFree(pSignerInfo);

        return false;

    }

    // Look for nested signature

    constexpr const char\* kOID\_NESTED\_SIGNATURE = "1.3.6.1.4.1.311.2.4.1";

    for (DWORD i = 0; i < pSignerInfo->UnauthAttrs.cAttr; i++)

    {

        if (strcmp(kOID\_NESTED\_SIGNATURE, pSignerInfo->UnauthAttrs.rgAttr[i].pszObjId) == 0)

        {

            HCRYPTMSG hMsg2 = pfnCryptMsgOpenToDecode(X509\_ASN\_ENCODING | PKCS\_7\_ASN\_ENCODING, 0, 0, NULL, NULL, NULL);

            if (hMsg2)

            {

                if (pfnCryptMsgUpdate(hMsg2,pSignerInfo->UnauthAttrs.rgAttr[i].rgValue->pbData,pSignerInfo->UnauthAttrs.rgAttr[i].rgValue->cbData,TRUE))

                {

                    dwSignerInfo = 0;

                    pfnCryptMsgGetParam(hMsg2, CMSG\_SIGNER\_INFO\_PARAM, 0, NULL, &dwSignerInfo);

                    if (dwSignerInfo != 0)

                    {

                        PCMSG\_SIGNER\_INFO pSignerInfo2 = (PCMSG\_SIGNER\_INFO)LocalAlloc(LPTR, dwSignerInfo);

                        if (pSignerInfo2)

                        {

                            if (pfnCryptMsgGetParam(hMsg2, CMSG\_SIGNER\_INFO\_PARAM, 0, (PVOID)pSignerInfo2, &dwSignerInfo))

                            {

                                CRYPT\_DATA\_BLOB c7Data;

                                c7Data.pbData = pSignerInfo->UnauthAttrs.rgAttr[i].rgValue->pbData;

                                c7Data.cbData = pSignerInfo->UnauthAttrs.rgAttr[i].rgValue->cbData;

                                auto hStore2 = pfnCertOpenStore(CERT\_STORE\_PROV\_PKCS7, X509\_ASN\_ENCODING | PKCS\_7\_ASN\_ENCODING, NULL, 0, &c7Data);

                                if (!hStore2)

                                {

                                    LocalFree(pSignerInfo2);

                                    return false;

                                }

                                CERT\_INFO CertInfo{};

                                PCCERT\_CONTEXT pCertContext = NULL;

                                // Search for the signer certificate in the temporary certificate store.

                                CertInfo.Issuer = pSignerInfo2->Issuer;

                                CertInfo.SerialNumber = pSignerInfo2->SerialNumber;

                                pCertContext = pfnCertFindCertificateInStore(hStore2,

                                    (X509\_ASN\_ENCODING | PKCS\_7\_ASN\_ENCODING),

                                    0,

                                    CERT\_FIND\_SUBJECT\_CERT,

                                    (PVOID)&CertInfo,

                                    NULL);

                                if (!pCertContext)

                                {

                                    LocalFree(pSignerInfo2);

                                    pfnCertCloseStore(hStore2, CERT\_CLOSE\_STORE\_FORCE\_FLAG);

                                    return false;

                                }

                                void\* decodedPublicKey{};

                                DWORD decodedPublicLength{};

                                if (pfnCryptDecodeObjectEx((PKCS\_7\_ASN\_ENCODING | X509\_ASN\_ENCODING),

                                    CNG\_RSA\_PUBLIC\_KEY\_BLOB,

                                    pCertContext->pCertInfo->SubjectPublicKeyInfo.PublicKey.pbData,

                                    pCertContext->pCertInfo->SubjectPublicKeyInfo.PublicKey.cbData,

                                    CRYPT\_ENCODE\_ALLOC\_FLAG,

                                    NULL,

                                    &decodedPublicKey,

                                    &decodedPublicLength))

                                {

                                    static uint8\_t s\_rsaStreamlinePublicKey[] =

                                    {

                                        0x52, 0x53, 0x41, 0x31, 0x00, 0x0c, 0x00, 0x00, 0x03, 0x00, 0x00, 0x00, 0x80, 0x01, 0x00, 0x00,

                                        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x00, 0x01, 0xc1, 0x8e, 0x40, 0xc3, 0xf5,

                                        0xa7, 0x01, 0x9a, 0x37, 0x6b, 0x47, 0xa8, 0x58, 0xe8, 0xbe, 0xe3, 0x55, 0x0a, 0xee, 0x0f, 0x0d,

                                        0x32, 0xaa, 0x12, 0xf9, 0x56, 0x7f, 0x5d, 0xfd, 0x82, 0x09, 0x33, 0x21, 0x42, 0xf2, 0xe8, 0x74,

                                        0x98, 0x51, 0xb3, 0x88, 0x74, 0xcd, 0x00, 0x6e, 0xb1, 0x08, 0x10, 0x4b, 0xf1, 0xda, 0xd6, 0x97,

                                        0x87, 0xd4, 0x9c, 0xb1, 0x13, 0xa8, 0xa2, 0x86, 0x15, 0x0e, 0xc1, 0xa5, 0x9c, 0xe5, 0x90, 0x9b,

                                        0xbe, 0x69, 0xdc, 0x6a, 0x82, 0xbe, 0xb4, 0x4b, 0x4b, 0xfa, 0x95, 0x8e, 0xc1, 0xfc, 0x2b, 0x61,

                                        0x95, 0xd1, 0x91, 0xed, 0xeb, 0x87, 0xe7, 0x09, 0x84, 0x05, 0x41, 0x03, 0xb0, 0x2d, 0xd4, 0x39,

                                        0x7f, 0x62, 0x06, 0x56, 0x33, 0x93, 0x7e, 0x77, 0x54, 0x06, 0x77, 0x2b, 0x75, 0x05, 0xbc, 0xeb,

                                        0x98, 0xea, 0xc0, 0xa2, 0xca, 0x98, 0x86, 0x0f, 0x10, 0x65, 0xde, 0x19, 0x2c, 0xa6, 0x1e, 0x93,

                                        0xb0, 0x92, 0x5d, 0x5f, 0x5b, 0x6f, 0x79, 0x6d, 0x2c, 0x76, 0xa6, 0x67, 0x50, 0xaa, 0x8f, 0xc2,

                                        0x4c, 0xf1, 0x08, 0xf7, 0xc0, 0x27, 0x29, 0xf0, 0x68, 0xf4, 0x64, 0x00, 0x1c, 0xb6, 0x28, 0x1e,

                                        0x25, 0xb8, 0xf3, 0x8a, 0xd1, 0x6e, 0x65, 0xa3, 0x61, 0x9d, 0xf8, 0xca, 0x4a, 0x41, 0x60, 0x80,

                                        0x62, 0xdf, 0x41, 0xa4, 0x8b, 0xdc, 0x97, 0xee, 0xeb, 0x64, 0x6f, 0xe4, 0x8f, 0x4b, 0xdf, 0x24,

                                        0x01, 0x80, 0xd9, 0xb4, 0x0a, 0xec, 0x0d, 0x3e, 0xb7, 0x76, 0xba, 0xe9, 0xe7, 0xde, 0x07, 0xdd,

                                        0x30, 0xc8, 0x4a, 0x14, 0x79, 0xec, 0x15, 0xed, 0x5c, 0xc6, 0xcc, 0xd4, 0xe6, 0x06, 0x3c, 0x42,

                                        0x92, 0x10, 0xf7, 0x7c, 0x80, 0x1e, 0x78, 0xd3, 0xb4, 0x9f, 0xc2, 0x3b, 0xa8, 0x7b, 0xa0, 0xe3,

                                        0x0c, 0xd9, 0xad, 0x2e, 0x09, 0x72, 0xe2, 0x8f, 0x54, 0x28, 0x87, 0x3c, 0xba, 0x7c, 0x97, 0x80,

                                        0xdc, 0x09, 0xb5, 0x12, 0x34, 0x78, 0x9a, 0x26, 0xd0, 0xa3, 0xa7, 0xa7, 0x1b, 0x25, 0x19, 0xe5,

                                        0x6e, 0xbe, 0xd7, 0x5a, 0x91, 0x32, 0xc4, 0xa9, 0x2f, 0xcc, 0xd5, 0x82, 0x4b, 0x5b, 0x9f, 0xad,

                                        0xf3, 0x2f, 0xed, 0x4f, 0x33, 0xe1, 0x50, 0x33, 0xd6, 0x90, 0x79, 0x22, 0xe5, 0x1c, 0xc7, 0x35,

                                        0xe7, 0x58, 0xe6, 0xb4, 0x8b, 0xc4, 0x28, 0x20, 0xec, 0xca, 0x70, 0xbb, 0x02, 0x1b, 0x48, 0xd8,

                                        0x84, 0x51, 0x24, 0x33, 0x2a, 0x08, 0xb1, 0x15, 0x4e, 0xbc, 0x88, 0xa5, 0xe1, 0x37, 0x76, 0x70,

                                        0xe6, 0xdf, 0x3f, 0x73, 0xfd, 0x0d, 0x8a, 0xd9, 0x0d, 0xa5, 0x35, 0xb2, 0xb4, 0x01, 0x42, 0x96,

                                        0xc4, 0xaa, 0x1c, 0xeb, 0x68, 0x62, 0x36, 0xbf, 0xef, 0x5e, 0x2a, 0x3d, 0x18, 0x91, 0x8b, 0x92,

                                        0x0a, 0x1e, 0xce, 0x98, 0x5b, 0x7b, 0x64, 0x42, 0x09, 0xb0, 0x1d

                                    };

                                    valid = decodedPublicLength == sizeof(s\_rsaStreamlinePublicKey) && memcmp(s\_rsaStreamlinePublicKey, decodedPublicKey, decodedPublicLength) == 0;

                                    LocalFree(decodedPublicKey);

                                }

                                pfnCertFreeCertificateContext(pCertContext);

                                pfnCertCloseStore(hStore2, CERT\_CLOSE\_STORE\_FORCE\_FLAG);

                            }

                            LocalFree(pSignerInfo2);

                        }

                    }

                }

                pfnCryptMsgClose(hMsg2);

            }

            break;

        }

    }

    LocalFree(pSignerInfo);

    pfnCryptMsgClose(hMsg);

    pfnCertCloseStore(hStore, CERT\_CLOSE\_STORE\_FORCE\_FLAG);

    return valid;

}

### 1

### 

# 유용한 함수

# 참고자료

# 신규 함수 참고

## bloaderapi.h

### FreeLibrary

#### <https://learn.microsoft.com/ko-kr/windows/win32/api/libloaderapi/nf-libloaderapi-freelibrary>

### LoadLibraryExW

#### <https://learn.microsoft.com/ko-kr/windows/win32/api/libloaderapi/nf-libloaderapi-loadlibraryexw>

## stddef.h

### offsetof

#### 참고자료: <https://learn.microsoft.com/ko-kr/cpp/c-runtime-library/reference/offsetof-macro?view=msvc-170>

#### 소스코드

##### #define offsetof(s,m) ((::size\_t)&reinterpret\_cast<char const volatile&>((((s\*)0)->m)))

## 서명 관련

### verifying <https://learn.microsoft.com/en-us/windows/win32/seccrypto/example-c-program--verifying-the-signature-of-a-pe-file>

### WinVerifyTrust - <https://learn.microsoft.com/ko-kr/windows/win32/api/wintrust/nf-wintrust-winverifytrust>

### CryptQueryObject <https://learn.microsoft.com/ko-kr/windows/win32/api/wincrypt/nf-wincrypt-cryptqueryobject>

### CryptMsgGetParam - <https://learn.microsoft.com/en-us/windows/win32/api/wincrypt/nf-wincrypt-cryptmsggetparam>

### CryptMsgOpenToDecode - <https://learn.microsoft.com/ko-kr/windows/win32/api/wincrypt/nf-wincrypt-cryptmsgopentodecode>

### CryptMsgUpdate - <https://learn.microsoft.com/ko-kr/windows/win32/api/wincrypt/nf-wincrypt-cryptmsgupdate>

### CertOpenStore - <https://learn.microsoft.com/ko-kr/windows/win32/api/wincrypt/nf-wincrypt-certopenstore>

### CertFindCertificateInStore - <https://learn.microsoft.com/ko-kr/windows/win32/api/wincrypt/nf-wincrypt-certfindcertificateinstore>

### CryptDecodeObjectEx - <https://learn.microsoft.com/ko-kr/windows/win32/api/wincrypt/nf-wincrypt-cryptdecodeobjectex>

### CertFreeCertificateContext - <https://learn.microsoft.com/ko-kr/windows/win32/api/wincrypt/nf-wincrypt-certfreecertificatecontext>

### CertCloseStore - <https://learn.microsoft.com/en-us/windows/win32/api/wincrypt/nf-wincrypt-certclosestore>

### CryptMsgClose - <https://learn.microsoft.com/ko-kr/windows/win32/api/wincrypt/nf-wincrypt-cryptmsgclose>

## ShellScalingApids.h

### SetProcessDpiAwareness - <https://learn.microsoft.com/ko-kr/windows/win32/api/shellscalingapi/nf-shellscalingapi-setprocessdpiawareness>

## dxgi1\_4.h

### IDXGISwapChain3

# 기존 프로젝트와 Streamline 병합 방법

## 기존 프로젝트 정합성 유지

### #ifdef 구문 활용 sl.interposer 선택 로직 구현

## 싱글턴 객체 연동

### sl::log 연동을 위해 dount log 관리 lib or dll 구현

## Initialize\_preDevice 구현

### 개선점

#### DeviceCreation Parameter 구조체 제작

### 

## 디버깅을 위해 .dll 연동 시스템 구현 (컴파일 후 \_sdk 폴더의 자료를 활용)

### 

## Hook 시 json 연동을 위해 dll 환경변수 설정 (정합한 위치를 참조하도록 프로젝트 세팅)

### 결과: json 파일을 참조하여 자동 세팅 완료

#### A screenshot of a computer AI-generated content may be incorrect.

#### 

## d3d12.dll 후킹

### 동작

#### d3d12.dll 내 함수를 불러온 후, streamline 내 함수로 교환

### 결과

#### A screenshot of a computer AI-generated content may be incorrect.

## 후킹 후 D3D12 셋업

### D3D12GetDebugInterface

### DXGIGetDebugInterface1

### CreateDXGIFactory1

### D3D12CreateDevice

## sl\_init 중 버그 발생

### 버그 발생 함수

//! Common JSON configuration containing OS version, driver version, GPU architecture, supported adapters, plugin's SHA etc.

//!

//! Each plugin calls this first then adds any additional information which is specific to it.

//!

void updateCommonEmbeddedJSONConfig(void\* jsonConfig, const common::PluginInfo& info)

{

    auto& ctx = (\*common::getContext());

    // Bit hacky but better than including big json header in common header

    json& config = \*(json\*)jsonConfig;

    // SL does not work on Win7, only Win10+

    sl::Version minOS(10, 0, 0);

    // Also some reasonably old min driver

    sl::Version minDriver(512, 15, 0);

    if (info.minOS)

    {

        minOS = info.minOS;

    }

    if (info.minDriver)

    {

        minDriver = info.minDriver;

    }

    PreferenceFlags preferenceFlags{};

    api::getContext()->parameters->get(sl::param::global::kPreferenceFlags, (uint64\_t\*)&preferenceFlags);

    // Provided to host

    Version detectedOS(ctx.caps->osVersionMajor, ctx.caps->osVersionMinor, ctx.caps->osVersionBuild);

    auto osSupported = minOS <= detectedOS;

    // Check if host wants us to bypass the OS check

    if (!osSupported && ((preferenceFlags & PreferenceFlags::eBypassOSVersionCheck) != 0))

    {

        osSupported = true;

        SL\_LOG\_WARN("Bypassing OS version check - detected OS v%s - required OS v%s", detectedOS.toStr().c\_str(), minOS.toStr().c\_str());

    }

    config["external"]["os"]["detected"] = detectedOS.toStr();

    config["external"]["os"]["required"] = minOS.toStr();

    config["external"]["os"]["supported"] = osSupported;

    // Detected driver version must be valid in order for us to test, but only if min driver is provided by the feature, if not any driver would work

    sl::Version detectedDriver(ctx.caps->driverVersionMajor, ctx.caps->driverVersionMinor, 0);

    auto driverSupported = minDriver <= detectedDriver || (!info.minDriver && !detectedDriver);

    config["external"]["driver"]["detected"] = detectedDriver.toStr();

    config["external"]["driver"]["required"] = minDriver.toStr();

    config["external"]["driver"]["supported"] = driverSupported;

    // Not supported on any adapter by default

    uint32\_t adapterMask = 0;

    for (uint32\_t i = 0; i < ctx.caps->gpuCount; i++)

    {

        std::string adapter = "gpu" + std::to\_string(i);

        config["external"]["adapters"][adapter]["detected"] = ctx.caps->adapters[i].architecture;

        config["external"]["adapters"][adapter]["required"] = info.minGPUArchitecture;

        config["external"]["adapters"][adapter]["supported"] = info.minGPUArchitecture <= ctx.caps->adapters[i].architecture;

        // Check OS always, driver only on NVDA architecture and for generic plugins min architecture will be 0 which will always pass

        if (osSupported && (driverSupported || ctx.caps->adapters[i].architecture == 0) && ctx.caps->adapters[i].architecture >= info.minGPUArchitecture)

        {

            adapterMask |= 1 << i;

        }

    }

    bool supported = adapterMask != 0;

    // Internal bits

    config["supportedAdapters"] = adapterMask;

    config["sha"] = info.SHA;

    std::vector<BufferType> tags;

    if (supported)

    {

        //! Report to host and log required tags if feature is supported

        //!

        Feature feature = config["id"];

        for (auto& t : info.requiredTags)

        {

            tags.push\_back(t.first);

            SL\_LOG\_VERBOSE("Registering required tag '%s' life-cycle '%s' for feature `%s`", getBufferTypeAsStr(t.first), getResourceLifecycleAsStr(t.second), getFeatureAsStr(feature));

        }

    }

    config["external"]["feature"]["tags"] = tags;

    // Only if feature is supported on at least one adapter

    if (supported)

    {

        ctx.needNGX |= info.needsNGX;

        ctx.needDX11On12 |= info.needsDX11On12;

        ctx.needDRS |= info.needsDRS;

    }

}

## 개선 방법

### sl.\*.dll 파일에 대해 nvidia 기본 파일 활용

### sl.\*.dll 파일을 -debug가 아닌 -production으로 컴파일 된 파일을 적용

#### 결과A black background with many small colored lines AI-generated content may be incorrect.

## D3D 객체 로드 순서

### DXGI Factory2

## 버그2 slSetD3DDevice 과정 중 nvngx.dll 로드 오류

### 에러 현상

#### 

#### 

## sl.dlss\_g.dll 라이브러리 로드 간 에러

### 우선 개인적으로 기본 sl.dlss\_g.dll 파일을 활용하도록 수정

## 결과

### 현재 2개의 swap chain을 활용하는 구조로 오류 발생. (하나의 swap chain 삭제 예정)

### 