UNREAL
ENGINE

언리얼 엔진 4 셰이더, 더 깊이 이해하기

셰이더 메모리를 줄이고 싶어요

부모 머티리얼에 Static Switch Parameter를 배치하면 캐시가 느는 건가요?

렌더링 패스에서 사용하는 셰이더랑 머티리얼이랑 무슨 관계죠?

머티리얼 에디터에 HLSL 코드 보는게 있던데, 정체가 뭔가요?

플러그인으로 글로벌 셰이더는 어떻게 추가하나요?

목표 – 더 깊이 이해하기

- 엔진 내부에서 벌어지는 일
  - USF와 UE4 셰이더의 관계
  - 작성한 머티리얼이 셰이더 컴파일러에 전달되기까지.

- 문제없는 셰이더 만들기
  - 머티리얼 노드에 없는 기능 추가하기
  - 플러그인으로 글로벌 셰이더 작성하기
  - 디버깅 팁

목차

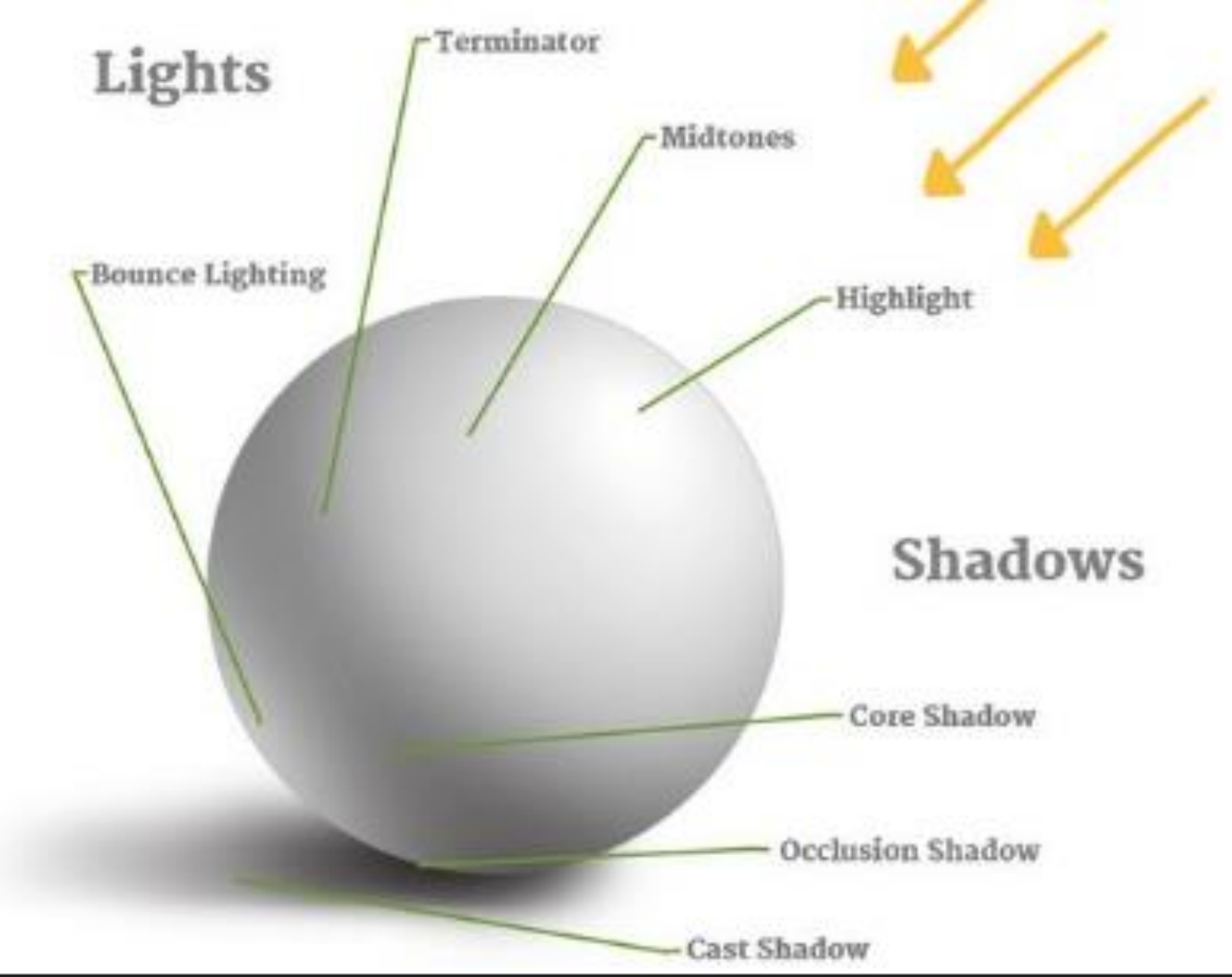
- UE4 셰이더 시스템
  - 셰이더
  - UE4 셰이더
  - FShader - FShaderTyp
  - 머티리얼과 UE4 셰이더
- UE4 셰이더 작성
- UE4 셰이더 디버깅

이건 이야기 안할 거에요

- Not to talk
  - 그래픽스 파이프라인
  - UE4 렌더러 분석
  - 렌더링 테크닉
  - HLSL 문법
  - 머티리얼 에디터 사용법

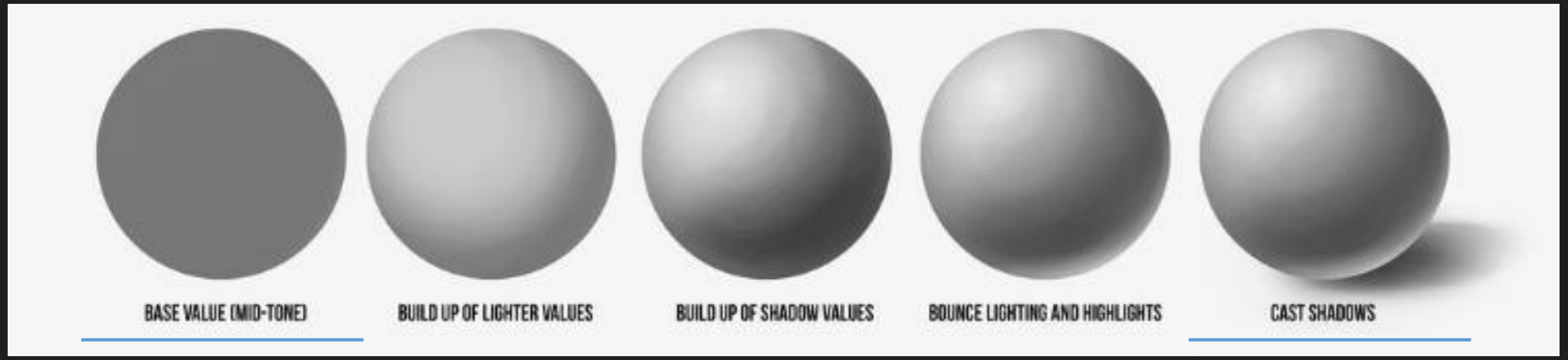
셰이더

What is shader?



셰이더

- Shading에 사용되는 작은 **프로그램**
  - Shading = 물체를 시각적으로 정의

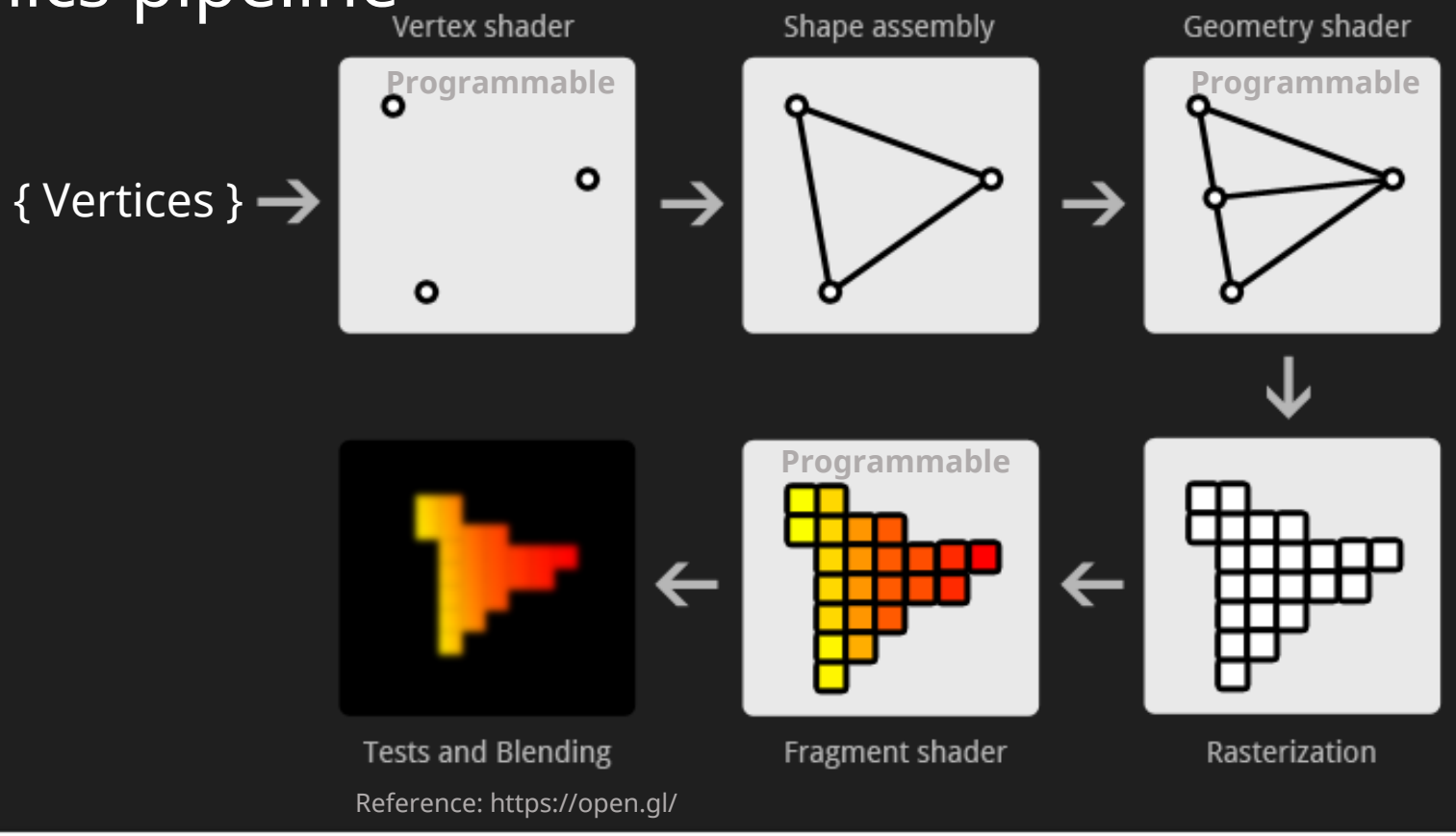


Reference: https://cgcookie.deviantart.com/

- (플랫폼 따라 다른) Shader Language로 작성.

셰이더

- Programmable graphics pipeline



Reference: https://open.gl/

- Programmable
  - = 변경 가능
  - = 우리가 만들 것

셰이더

- Programmable Shader 구분 (Shading Frequency)
  - Vertex Shader
  - Hull Shader
  - Domain Shader
  - Geometry Shader
  - Pixel / Fragment Shader
  - Compute Shader

셰이더

ex) Pixel Shader
on WebGL

- 눈여겨 볼 만한 것
  - Shader Inputs
  - Entry point

Shader Inputs

```
uniform vec3      iResolution;           // viewport resolution (in pixels)
uniform float     iTime;                 // shader playback time (in seconds)
uniform float     iTimeDelta;            // render time (in seconds)
uniform int       iFrame;                // shader playback frame
uniform float     iChannelTime[4];       // channel playback time (in seconds)
uniform vec3      iChannelResolution[4]; // channel resolution (in pixels)
uniform vec4      iMouse;                // mouse pixel coords. xy: current (if MLB down), zw: click
uniform samplerXX iChannel0..3;          // input channel. XX = 2D/Cube
uniform vec4      iDate;                 // (year, month, day, time in seconds)
uniform float     iSampleRate;           // sound sample rate (i.e., 44100)
```

```
11 //
12 // The trees are really cheap (ellipsoids with noise), but they kind of do the job in
13 // distance and low image resolutions.
14 //
15 // I used some cheap reprojection technique to smooth out the render, although it creates
16 // halows and blurs the image way too much (I don't the time now to do the tricks used in
17 // TAA). Enable the STATIC_CAMERA define to see a sharper image.
18 //
19 // Lastly, it runs very slow, so I had to make a youtube capture, sorry for that!
20 //
21 // https://www.youtube.com/watch?v=VqYROPZrDeU
22
23
24
25
26 void mainImage( out vec4 fragColor, in vec2 fragCoord )
27 {
28     vec2 p = fragCoord/iResolution.xy;
29
30     vec3 col = texture( iChannel0, p ).xyz;
31     //vec3 col = texelFetch( iChannel0, ivec2(fragCoord-0.5), 0 ).xyz;
32
33     col *= 0.5 + 0.5*pow( 16.0*p.x*p.y*(1.0-p.x)*(1.0-p.y), 0.05 );
34
35     fragColor = vec4( col, 1.0 );
36 }
37
```

Reference: https://www.shadertoy.com/view/4ttSWf

셰이더

- 방금 그 셰이더를 활용한 WebGL 샘플
  - 잘 다루면 Pixel Shader만으로 이런 것도 가능
  - https://www.shadertoy.com/view/4ttSWf
  - Created by Inigo Quilez

UE4 셰이더

Shaders in UE4

USH / USF

- “.hlsl”이 아닌,
- HLSL 문법을 따르는 셰이더 파일
  - Unreal Shader Headers
  - Unreal Shader Format

- Multi-platform 지원
  - 원할때마다 타겟 플랫폼에서 돌아가는 셰이더로 변경!!
  - 즉, 셰이더의 “크로스 컴파일”이 된다는 것

USH / USF

- USH (in UE4.17+)
  - Include 되는 용도
  - 주로 Public 폴더
  - Entry point 없음 ➔ 함수 라이브러리나 컴파일 Definition

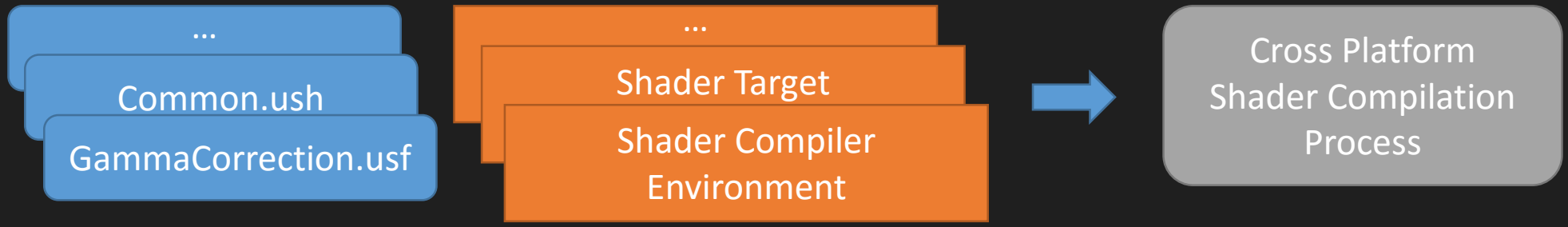
- USF
  - 주로 USH를 Include
  - 주로 Private 폴더
  - Entry point 있음

USH / USF

- Entry Points
  - MainVS(..) / MainPS(..)
- Inputs
  - half InverseGamma
- #include USHs
- #if, #endif, #define, …
- …

```
/*=============================================================================
    GammaCorrection.usf: Gamma correcting the scene color buffer
=============================================================================*/

#include "Common.ush"

// vertex shader entry point
void MainVS(
    in float4 InPosition : ATTRIBUTE0,
    in float2 InTexCoord : ATTRIBUTE1,
    out float2 OutTexCoord : TEXCOORD0,
    out float4 OutPosition : SV_POSITION
    )
{
    DrawRectangle(InPosition, InTexCoord, OutPosition, OutTexCoord);
}

half3 ColorScale;
half4 OverlayColor;
half InverseGamma;

half3 TonemapAndGammaCorrect(half3 LinearColor) { ... }

// pixel shader entry point
void MainPS(float2 UV : TEXCOORD0, out float4 OutColor : SV_Target0)
{
    half4 LinearColor = Texture2DSample(SceneColorTexture, SceneColorTextureSampler, UV);
    half3 LDRColor = TonemapAndGammaCorrect(LinearColor.rgb);

    // blend with custom LDR color, used for Fade track in Matinee
    LDRColor = lerp(LDRColor * ColorScale, OverlayColor.rgb, OverlayColor.a);

    // RETURN_COLOR not needed unless writing to SceneColor
    OutColor = float4(LDRColor, LinearColor.a);
}
```

USH / USF – to shader compiler



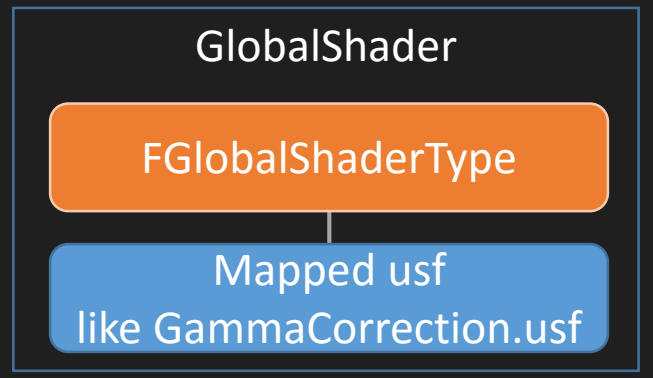
- Shader Target
  - Shader Frequency
  - Platform
- Shader Compiler Environment
  - Compiler Flags
  - Definitions
  - ...
- 그 후엔 Cross compile의 영험함을 즐기시면...

FShader – FShaderType

FShaderType

- USF와 C++클래스를 맵핑
- 엄밀히 이야기하면
  - FShaderType == Meta type
  - FShader 클래스의 정보 선언

  (※ FShaderType != Fshader)
- 매크로로 선언/구현




```
class FGammaCorrectionPS : public FGlobalShader
{
    DECLARE_SHADER_TYPE(FGammaCorrectionPS,Global);
```

```
IMPLEMENT_SHADER_TYPE(,FGammaCorrectionPS,TEXT("/Engine/Private/GammaCorrection.usf"),TEXT("MainPS"),SF_Pixel);
```

- 언리얼 엔진에 USF 등록

FShaderType

- Meta type이 FShader에 구현 요구하는 주요 함수
  - *Constructors w/ SerializedInstance or CompiledInstance*
  - *ShouldCache()*
    - 해당 FShader의 캐시 여부 결정
    - > 내가 만든 usf를 컴파일러에게 넘길지 말지
  - *ModifyCompilationEnvironment()*
    - 컴파일러에게 전달될 Shader Compiler Environment 생성시 활용
    - > 내가 만든 usf가 원하는 셰이딩 연산을 할수 있도록

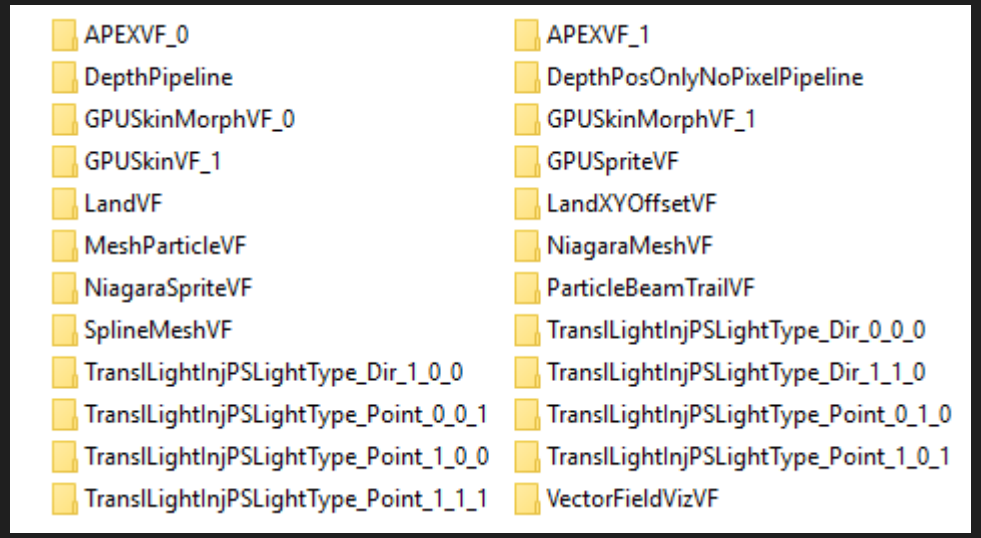
FShaderType

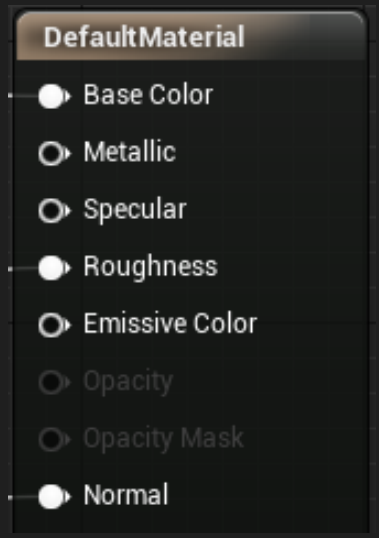
- 3가지 유형 존재
  - FGlobalShaderType, FMaterialShaderTyp, FMeshMaterialShderType
- 셰이딩시 어떤 정보가 필요한가
  - 메시 특성
  - 머티리얼 특성

	글로벌 셰이더	머티리얼 셰이더	메시 머티리얼 셰이더
메시 특성	X	X	O
머티리얼 특성	X	O	O

FShaderType

- 메시 특성
  - Vertex Factory로 추상화
  - 작성 인터페이스 존재 X
- 머티리얼 특성
  - 머티리얼을 사용하나
  - 머티리얼 에디터로 작성

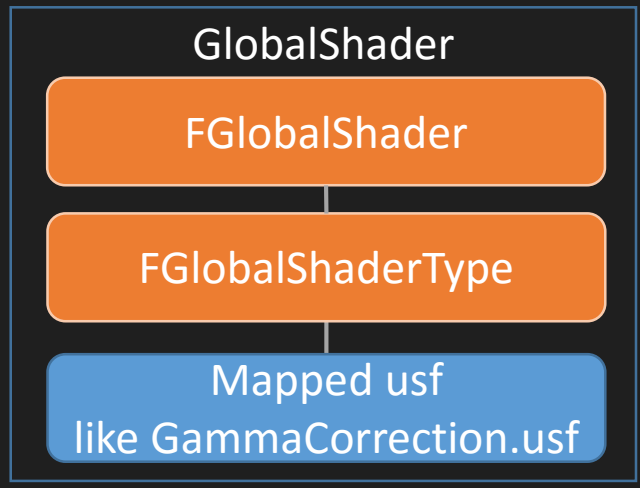




FShader

- FShaderType의 컴파일된 셰이더 인스턴스

- FShaderType처럼, FShader도 3가지 유형 존재
  - FGlobalShader, FMaterialShader, FMeshMaterialShader



- 보통 둘을 함께 생각함.
  - 글로벌 셰이더, 머티리얼 셰이더, 메시머티리얼 셰이더라고 칭함
  - FGlobalShader를 상속한 FGammaCorrectionPS ➔ 감마커렉션 픽셀셰이더

FShader

- FShader 주요 멤버
  - **FShaderResource**
    - 셰이딩에 필요한 정보
    - binary shader / RHI resource
  - *Serialize()*
    - Serialization/deserialization 처리
- FGlobalShader 주요 함수
  - *Default constructor*
  - *Initialization constructor*
  - *ShouldCache()*
  - *ModifyCompilationEnvironment()*
  - *SetParameters()*
    - 바인드된 파라미터의 값 결정

FShader

- FMaterialShader 주요 함수
  - *Default constructor*
  - *Initialization constructor*
  - *ShouldCache()*
  - *ModifyCompilationEnvironment()*
  - *SetParameters()* with Material
- FMeshMaterialShader 주요 함수
  - *Default constructor*
  - *Initialization constructor*
  - *ShouldCache()*
  - *ModifyCompilationEnvironment()*
  - *SetParameters()* with Material
  - *SetMesh()*

※ 꼭 *SetParameters()* 함수를 사용해야하는 건 아닙니다.

예시 – FGlobalShader

- FGammaCorrectionPS

GammaCorrection.cpp

FGlobalShader

class FGlobalShader : public FShader{...}

```
19
20      /** Encapsulates the gamma correction pixel shader. */
21      class FGammaCorrectionPS : public FGlobalShader
22      {
23          DECLARE_SHADER_TYPE(FGammaCorrectionPS,Global);
24
25          static bool ShouldCache(EShaderPlatform Platform)
26          {
27              return true;
28          }
29
30          /** Default constructor. */
31          FGammaCorrectionPS() {}
```

예시 – FGlobalShader

- FGammaCorrectionPS

```
33  public:
34
35      FShaderResourceParameter SceneTexture;
36      FShaderResourceParameter SceneTextureSampler;
37      FShaderParameter InverseGamma;
38      FShaderParameter ColorScale;
39      FShaderParameter OverlayColor;
40
41      /** Initialization constructor. */
42      FGammaCorrectionPS(const ShaderMetaType::CompiledShaderInitializerType& Initializer)
43          : FGlobalShader(Initializer)
44      {
45          SceneTexture.Bind(Initializer.ParameterMap,TEXT("SceneColorTexture"));
46          SceneTextureSampler.Bind(Initializer.ParameterMap,TEXT("SceneColorTextureSampler"));
47          InverseGamma.Bind(Initializer.ParameterMap,TEXT("InverseGamma"));
48          ColorScale.Bind(Initializer.ParameterMap,TEXT("ColorScale"));
49          OverlayColor.Bind(Initializer.ParameterMap,TEXT("OverlayColor"));
50      }
```

예시 – FGlobalShader

- FGammaCorrectionPS

```
52      // FShader interface.
53      virtual bool Serialize(FArchive& Ar) override
54      {
55          bool bShaderHasOutdatedParameters = FGlobalShader::Serialize(Ar);
56
57          Ar << SceneTexture << SceneTextureSampler << InverseGamma << ColorScale << OverlayColor;
58          return bShaderHasOutdatedParameters;
59      }
60  };
61
```

```
IMPLEMENT_SHADER_TYPE(,FGammaCorrectionPS,TEXT("/Engine/Private/GammaCorrection.usf"),TEXT("MainPS"),SF_Pixel);
```

다른 FShaderType들의 예제는?

- 머티리얼 특성을 활용하는 다른 셰이더 타입들은
  - “머티리얼과 UE4 셰이더” 주제 후...

머티리얼과 UE4 셰이더

Materials as UE4 Shader

머티리얼과 UE4 셰이더

- 머티리얼 = 머티리얼 특성을 정의
  - Material Properties: ex) Base Color, Metalic
  - 어느 셰이더에 사용되나
    - Ex) Usage의 Used With Skeletal Mesh ➔ 메시 특성을 지정
    - Ex) Material Domain == Light Function ➔ 메시 특성 불필요

	글로벌 셰이더	머티리얼 셰이더	메시 머티리얼 셰이더
메시 특성	X	X	O
머티리얼 특성	X	O	O

머티리얼과 UE4 셰이더

- 머티리얼 에디터에서 “적용”을 누른다면?

- 2가지 단계 진행 후
  - Translation process
  - CacheResource process

- 준비된 USH와 USF 파일이 셰이더 컴파일러에 넘겨진다.

머티리얼과 UE4 셰이더

- 1단계: Translation process = 머티리얼 노드 해석
  - Material Node 해석
  - MaterialTemplate.ush에 해석된 노드 삽입
  - ➔ /Engine/Generated/Material.ush 생성

MaterialTemplate.ush

```
1287  half3 GetMaterialEmissiveRaw(FPixelMaterialInputs PixelMaterialInputs)
1288  {
1289      return PixelMaterialInputs.EmissiveColor;
1290  }
1291
1292  half3 GetMaterialEmissive(FPixelMaterialInputs PixelMaterialInputs)
1293  {
1294      half3 EmissiveColor = GetMaterialEmissiveRaw(PixelMaterialInputs);
1295  #if !MATERIAL_ALLOW_NEGATIVE_EMISSIVECOLOR
1296      EmissiveColor = max(EmissiveColor, 0.0f);
1297  #endif
1298      return EmissiveColor;
1299  }
1300
```

MaterialTemplate.ush

```
110  /**
111   * Parameters calculated from the pixel material inputs.
112   */
113  struct FPixelMaterialInputs
114  {
115  %s
116  };
117
```

머티리얼과 UE4 셰이더

- 1단계: Translation process = 머티리얼 특성 생성
  ※ Generated Material.ush 확인: 머티리얼 에디터 ➔ 창 ➔ HLSL 코드

MaterialTemplate.ush

```
110  /**
111   * Parameters calculated from the pixel material inputs.
112   */
113  struct FPixelMaterialInputs
114  {
115  %s
116  };
117
```

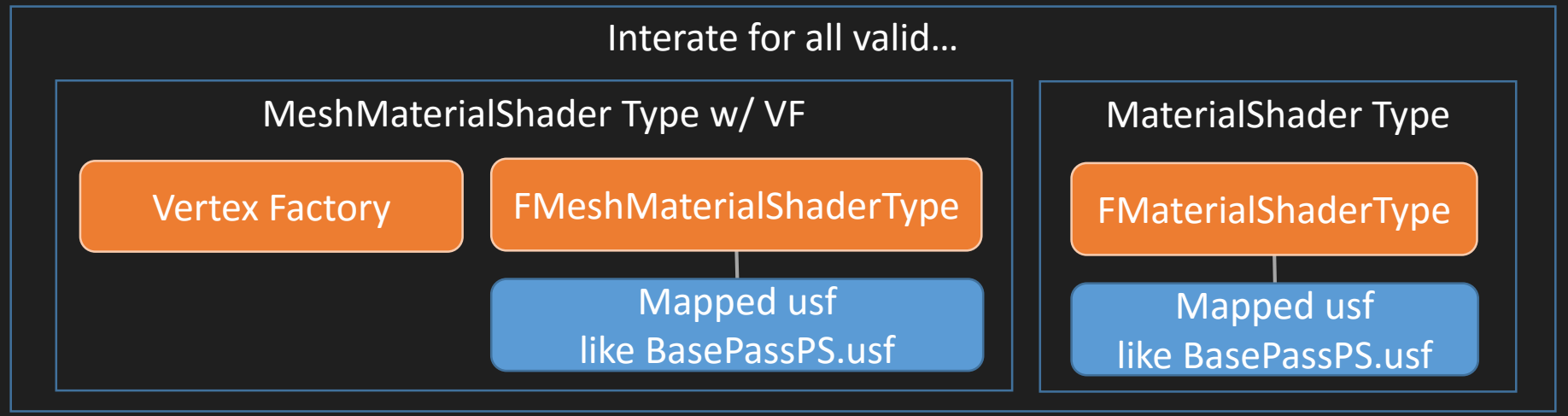
GeneratedMaterial.ush

```
110  /**
111   * Parameters calculated from the pixel material inputs.
112   */
113  struct FPixelMaterialInputs
114  {
115      MaterialFloat3 EmissiveColor;
116      MaterialFloat Opacity;
117      MaterialFloat OpacityMask;
118      MaterialFloat3 BaseColor;
```

머티리얼과 UE4 셰이더

- 2단계: CacheResource process = Generated USH를 사용하는 함께 컴파일될 USF 찾기
  - 대상 Quality Level과 Platform에 한해,
  - 엔진이 알고 있는 글로벌 셰이더를 제외한 모든 FShader에 대해 캐시 시도

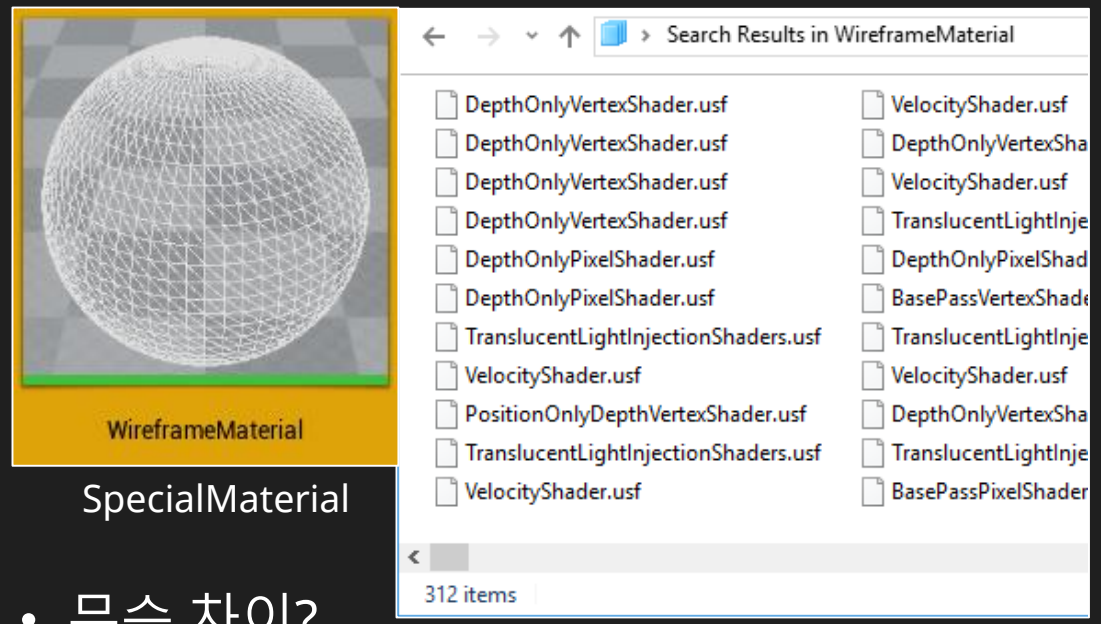
  - 최대 캐시 생성 수 = 모든 VF 수 x 모든 메시 머터리얼 셰이더 수 + 모든 머터리얼 셰이더 수 + @
    - ShouldCacheMeshShader(...) == TRUE면, 캐시
    - ShouldCacheMaterialShader(...) == TRUE면, 캐시



머티리얼과 UE4 셰이더

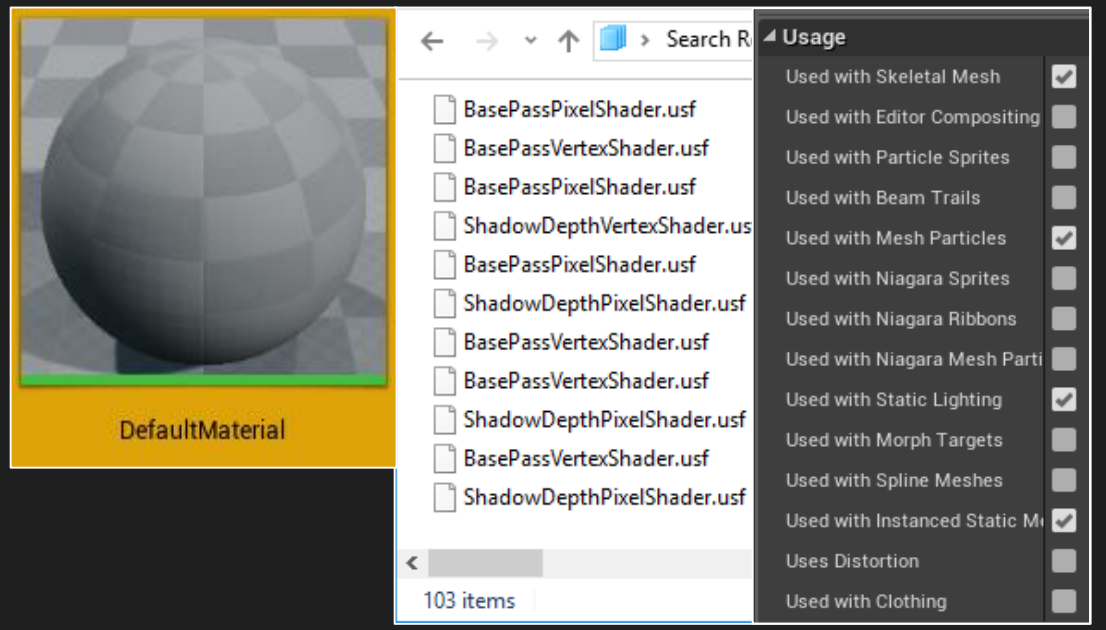
※ 예제 – PCD3D_SM5, High Quality

- WireframeMaterial : 312개



SpecialMaterial

- DefaultMaterial : 103개



- 무슨 차이?
  - 캐시 가능한 것들의 수 ➔ ShouldCache()

머티리얼과 UE4 셰이더

- Static Parameters
  - 머티리얼 인스턴스에 영향
  - 종류: StaticSwitchParameter / StaticComponentMask / TerrainLayerWeight

  ex) 부모 머티리얼의 Static Switch Pameter 기본값 = TRUE
  - 자식A가 FALSE로 오버라이드 했다면... 셰이더 추가 생성
  - 자식B가 FALSE로 오버라이드 했다면... 이미 있으니 추가 X

예시 – FMaterialShader

- PostProcessMaterialShaders.usf

```
7   #include "Common.ush"
8   #include "/Engine/Generated/Material.ush"
9
10  #if (FEATURE_LEVEL > FEATURE_LEVEL_ES3_1)
11
12  void MainVS(
13      in float4 InPosition : ATTRIBUTE0,
14      out float4 OutPosition : SV_POSITION
15      )
16  {
17      DrawRectangle(InPosition,  OutPosition);
18  }
19
20  void MainPS(
21      in float4 SvPosition : SV_Position,     // after all interpolators
22      out float4 OutColor : SV_Target0
23      )
24  {
```

예시 – FMaterialShader

- PostProcessMaterialShaders.usf

```
20  void MainPS(
21      in float4 SvPosition : SV_Position,     // after all interpolators
22      out float4 OutColor : SV_Target0
23      )
24  {
25      ResolvedView = ResolveView();
26      FMaterialPixelParameters Parameters = MakeInitializedMaterialPixelParameters();
27      FPixelMaterialInputs PixelMaterialInputs;
28
29      // can be optimized
30      float2 ScreenUV = SvPositionToBufferUV(SvPosition);
31
32  #if NUM_MATERIAL_TEXCOORDS
33      for(int CoordinateIndex = 0;CoordinateIndex < NUM_MATERIAL_TEXCOORDS;CoordinateIndex++)
34      {
35          Parameters.TexCoords[CoordinateIndex] = ScreenUV;
36      }
37  #endif
```

예시 – FMaterialShader

- PostProcessMaterialShaders.usf

```
38
39      Parameters.VertexColor = 1;
40
41      SvPosition.z = LookupDeviceZ(ScreenUV);
42      SvPosition.z = max(SvPosition.z, 1e-18);
43
44      // fill out other related material parameters
45      CalcMaterialParametersPost(Parameters, PixelMaterialInputs, SvPosition, true);
46
47      // Grab emissive colour as output
48  #if MATERIAL_OUTPUT_OPACITY_AS_ALPHA
49      const float Alpha = GetMaterialOpacity(PixelMaterialInputs);
50  #else
51      const float Alpha = 1.0f;
52  #endif
53      OutColor = float4(GetMaterialEmissive(PixelMaterialInputs), Alpha );
54  }
55
```

예시 – FMaterialShader

- FPostProcessMaterialPS

```
static bool ShouldCachePostProcessMaterial(EPostProcessMaterialTarget MaterialTarget, EShaderPlatform Platform, const FMaterial* Material)
{
    if (Material->GetMaterialDomain() == MD_PostProcess)
    {
        switch (MaterialTarget)
        {
        case EPostProcessMaterialTarget::HighEnd:
            return IsFeatureLevelSupported(Platform, ERHIFeatureLevel::SM4);
        case EPostProcessMaterialTarget::Mobile:
            return IsMobilePlatform(Platform) && IsMobileHDR();
        }
    }

    return false;
}
```

예시 – FMaterialShader

- FPostProcessMaterialPS

```
132
133     void SetParameters(FRHICommandList& RHICmdList, const FRenderingCompositePassContext& Context, const FMaterialRenderPro
134     {
135         const FPixelShaderRHIParamRef ShaderRHI = GetPixelShader();
136
137         FMaterialShader::SetParameters(RHICmdList, ShaderRHI, Material, *Material->GetMaterial(Context.View.GetFeatureLevel
138         PostprocessParameter.SetPS(ShaderRHI, Context, TStaticSamplerState<SF_Point,AM_Clamp,AM_Clamp,AM_Clamp>::GetRHI());
139     }
140
141     virtual bool Serialize(FArchive& Ar) override { ... }
147
148 private:
149     FPostProcessPassParameters PostprocessParameter;
150 };
151
152 typedef FPostProcessMaterialPS<EPostProcessMaterialTarget::HighEnd> FFPostProcessMaterialPS_HighEnd;
153 typedef FPostProcessMaterialPS<EPostProcessMaterialTarget::Mobile> FPostProcessMaterialPS_Mobile;
154
155 IMPLEMENT_MATERIAL_SHADER_TYPE(template<>,FFPostProcessMaterialPS_HighEnd,TEXT("/Engine/Private/PostProcessMaterialShaders.
156 IMPLEMENT_MATERIAL_SHADER_TYPE(template<>,FPostProcessMaterialPS_Mobile,TEXT("/Engine/Private/PostProcessMaterialShaders.us
```

예시 – FMeshMaterialShader

- 타입별 차이가 조금 있을 뿐. ∴ 생략

- 결국 중요한 것은 동일합니다.
  - USF, FShaderType, 그리고 Fhader
  - FShaderType이 요구한 함수들
  - 셰이더에 바인드되는 ShaderParameter들
  - SetParameter() 등으로 ShaderParameter에 값 넣어주기

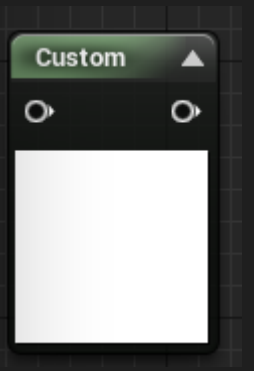
UE4 셰이더 시스템 – 마무리

- 주요 포인트 요약
  - 셰이더 = 셰이딩을 위한 작은 프로그램
  - 언리얼엔진에선 USF가 필요할때 셰이더로 변환.
  - USF는 셰이더 클래스와 연결. 총 3가지 타입.
  - 머티리얼은 Translation되어 임시로 USH파일 형태로 변경.
  - 자동 생성되었든 아니든 준비된 여러 USH들과 Entry가 있는 USF를 함께 컴파일러에 전달.
  - 크로스 컴파일 되어 여러 플랫폼에서 셰이딩

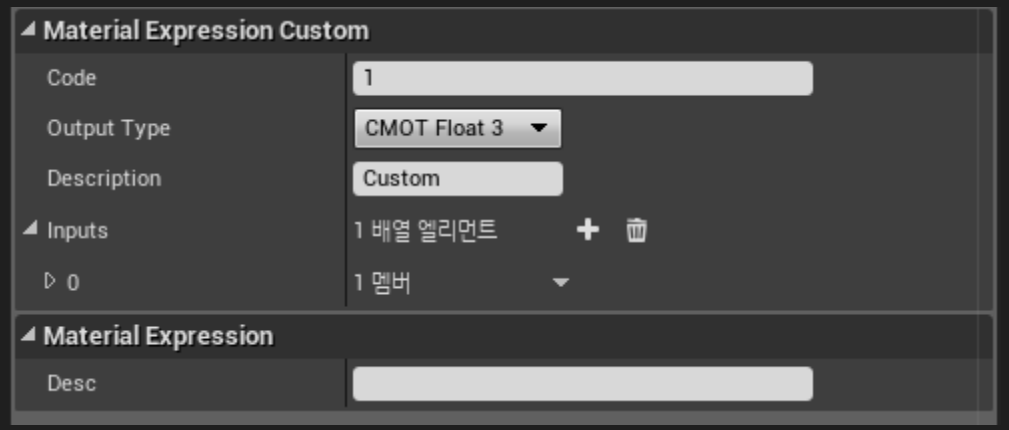
UE4 셰이더 작성

머티리얼

- 커스텀 노드
  - 머티리얼 에디터에서 추가
  - 노드로 처리가 복잡한, 혹은 특별한 기능을 추가하고자 할때.



- Translation process 중 코드 삽입



※ 주의사항: 노드 기반의 최적화 프로세스에서 제외됨.

글로벌 셰이더 포 플러그인

- 4.17에서 추가된 기능 – 실험단계
  - 핵심: 셰이더 컴파일러와 셰이더 디렉토리 파일 구조 변경

- 글로벌 셰이더를 플러그으로 추가 가능
  - 모듈화. 재사용성 증가.
  - 엔진 재컴파일 불필요.

글로벌 셰이더 포 플러그인

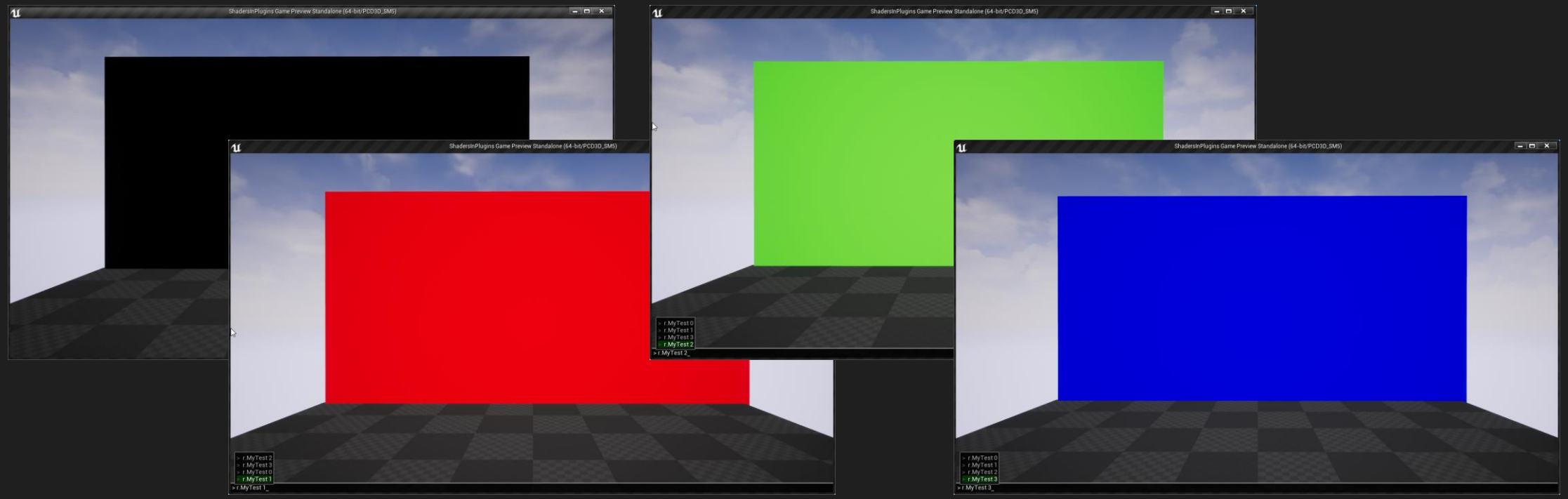
- 셰이더 검색시
  - First look in Engine/Shaders as usual
  - Then in project's Shader folder
  - Then in all enabled plugin Shader folders

- 다른 플러그인의 USH를 마음대로 #Include 가능
  - FShader와 USF 맵핑시 Full-path로 지정
  - .uproject처럼 플러그인간 종속성 관리 필요.

※주의: 플러그인으로 렌더링 패스의 추가/삭제가 가능해졌다는 뜻이 아님.

글로벌 셰이더 포 플러그인

- 예제
  - 렌더타겟에 특정 컬러를 그리는 간단한 플러그인



글로벌 셰이더 포 플러그인

- 1단계: 개발환경 세팅
  - 셰이더 개발 모드 켜기

```
34   ; Uncomment to get detailed logs on shader compiles and the opportunity to retry on errors
35   r.ShaderDevelopmentMode=1
```

  - 플러그인 마법사로 플러그인 추가하기
  - <생성된플러그인>.uplugin 파일에 로딩 페이즈 바꿔주기

```
17       "Modules": [
18           {
19               "Name": "HelloShadersInPlugin",
20               "Type": "Developer",
21               "LoadingPhase" : "PostConfigInit"
22           }
```

글로벌 셰이더 포 플러그인

- 2단계: USH / USF 작성하기
  - #pragma once
  - #include시 Virtual File path 사용
    - 엔진에 존재하는 셰이더: #include “/Engine/<FilePath>” , /Engine/Shaders/<FilePath>가 아닌.
    - 다른 플러그인에 존재하는 셰이더: #include “/Plugin/<PluginName>/<FilePath>”
  - #include “/Engine/Public/Platform.ush” to cross-compile

MyTest.usf

```
1   #pragma once
2
3   //#include "/Engine/Shaders/Public/ShaderVersion.
4   #include "/Engine/Public/ShaderVersion.ush"
5   #include "/Engine/Public/Platform.ush"
6
7
8   void MainVS(
```

```
9   void MainVS(
10      in float4 InPosition : ATTRIBUTE0,
11      out float4 Output : SV_POSITION )
12  {
13      Output = InPosition;
14  }
15
16  float4 MyColor;
17  float4 MainPS() : SV_Target0
18  {
19      return MyColor;
20  }
```

글로벌 셰이더 포 플러그인

- 3단계: GlobalShader 작성 – VS / PS

```
class FSimplePaintColorVS : public FGlobalShader
{
    DECLARE_SHADER_TYPE(FSimplePaintColorVS, Global);
};

IMPLEMENT_SHADER_TYPE(, FSimplePaintColorVS, TEXT("/Plugin/HelloShadersInPlugin/Private/MyTest.usf"), TEXT("MainVS"), SF_
```

```
class FSimplePaintColorPS : public FGlobalShader
{
    DECLARE_SHADER_TYPE(FSimplePaintColorPS, Global);
};

IMPLEMENT_SHADER_TYPE(, FSimplePaintColorPS, TEXT("/Plugin/HelloShadersInPlugin/Private/MyTest.usf"), TEXT("MainPS"), SF_Pixel)
```

※ PS에 컬러 정보 바인드

```
    void SetColor(FRHICommandList& RHICmdList, const FLinearColor& Color)
    {
        SetShaderValue(RHICmdList, GetPixelShader(), MyColorParameter, Color);
    }

private:
    FShaderParameter MyColorParameter;
```

글로벌 셰이더 포 플러그인

- 4단계: Rendering Function 생성
  - Input: TargetColor, RenderTarget Texture
    - 이 렌더타겟은 블루프린트로 넣어줌
  - Output: Updated RenderTarget Texture

- 주요 로직
  - 사용할 글로벌 셰이더 가져오기

```
TShaderMap<FGlobalShaderType>* GlobalShaderMap = GetGlobalShaderMap(
TShaderMapRef< FSimplePaintColorVS > VertexShader(GlobalShaderMap);
TShaderMapRef< FSimplePaintColorPS > PixelShader(GlobalShaderMap);
```

  - 버퍼에 색상 정보 넣어주기

```
PixelShader->SetColor(RHICmdList, TargetColor);
```

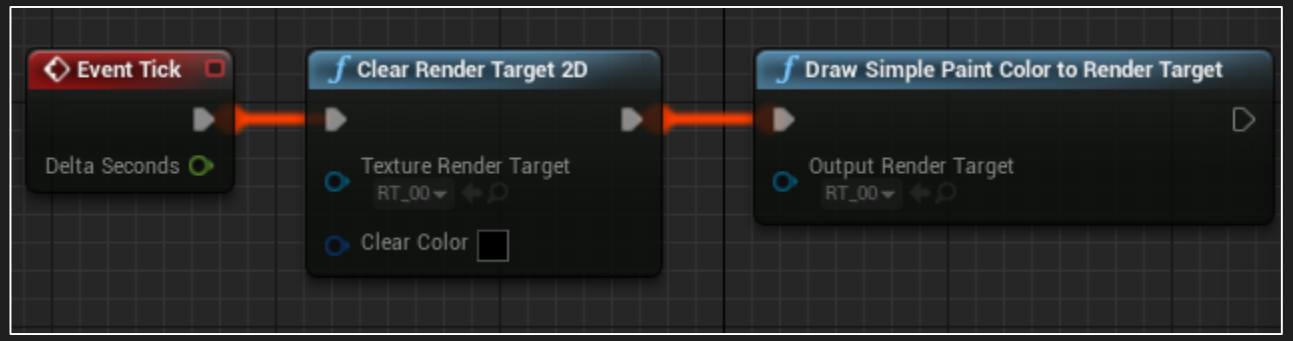
```
static void DrawSimplePaintColor_RenderThread(
    FRHICommandListImmediate& RHICmdList,
    const FLinearColor& TargetColor,
    FTextureRenderTargetResource* OutRenderTarget,
    ERHIFeatureLevel::Type FeatureLevel)
{
    check(IsInRenderingThread());

    // Set render target.
    SetRenderTarget(
        RHICmdList,
        OutRenderTarget->GetRenderTargetTexture(),
        FTextureRHIRef(),
        ESimpleRenderTargetMode::EUninitializedColorAndDepth,
        FExclusiveDepthStencil::DepthNop_StencilNop);

    RHICmdList.SetViewport(0, 0, 0.f, OutRenderTarget->GetSizeX(), OutRenderTarget->GetSizeY(), 1.f);

    TShaderMap<FGlobalShaderType>* GlobalShaderMap = GetGlobalShaderMap(FeatureLevel);
    TShaderMapRef< FSimplePaintColorVS > VertexShader(GlobalShaderMap);
    TShaderMapRef< FSimplePaintColorPS > PixelShader(GlobalShaderMap);

    // Set the graphic pipeline state.
    FGraphicsPipelineStateInitializer GraphicsPSOInit;
    RHICmdList.ApplyCachedRenderTargets(GraphicsPSOInit);
    GraphicsPSOInit.DepthStencilState = TStaticDepthStencilState<false, CF_Always>::GetRHI();
    GraphicsPSOInit.BlendState = TStaticBlendState<>::GetRHI();
    GraphicsPSOInit.RasterizerState = TStaticRasterizerState<>::GetRHI();
    GraphicsPSOInit.PrimitiveType = PT_TriangleList;
    GraphicsPSOInit.BoundShaderState.VertexDeclarationRHI = GetVertexDeclarationFVector4();
    GraphicsPSOInit.BoundShaderState.VertexShaderRHI = GETSAFERHISHADER_VERTEX(*VertexShader);
    GraphicsPSOInit.BoundShaderState.PixelShaderRHI = GETSAFERHISHADER_PIXEL(*PixelShader);
    SetGraphicsPipelineState(RHICmdList, GraphicsPSOInit);

    RHICmdList.SetViewport(0, 0, 0.f,OutRenderTarget->GetSizeX(), OutRenderTarget->GetSizeY(), 1.f);

    PixelShader->SetColor(RHICmdList, TargetColor);

    RHICmdList.DrawPrimitive(PT_TriangleStrip, 0, 2, 1);

    // Resolve render target.
    RHICmdList.CopyToResolveTarget(
        OutRenderTarget->GetRenderTargetTexture(),
        OutRenderTarget->TextureRHI,
        false, FResolveParams());
}
```

글로벌 셰이더 포 플러그인

- 5단계: 컨텐츠 레벨에서 다루기
  - BlueprintCallable 생성

```
UFUNCTION(BlueprintCallable, Category = "HelloShaders", meta = (WorldContex
static void DrawSimplePaintColorToRenderTarget(
    const UObject* WorldContextObject,
    class UTextureRenderTarget2D* OutputRenderTarget
    );
```

  - 렌더타겟 넘겨주기



글로벌 셰이더 포 플러그인

- 어렵지 않게 플러그인에 셰이더 추가 가능!!

- 한계
  - 아직 드라마틱한 변하는 안됩니다.
    - Ex) 머티리얼 에디터에서 사용할 수 있는 새로운 모델을 플러그인으로 생성!!
    - 다만... “아직은”

UE4 셰이더 디버깅

UE4 셰이더 디버깅

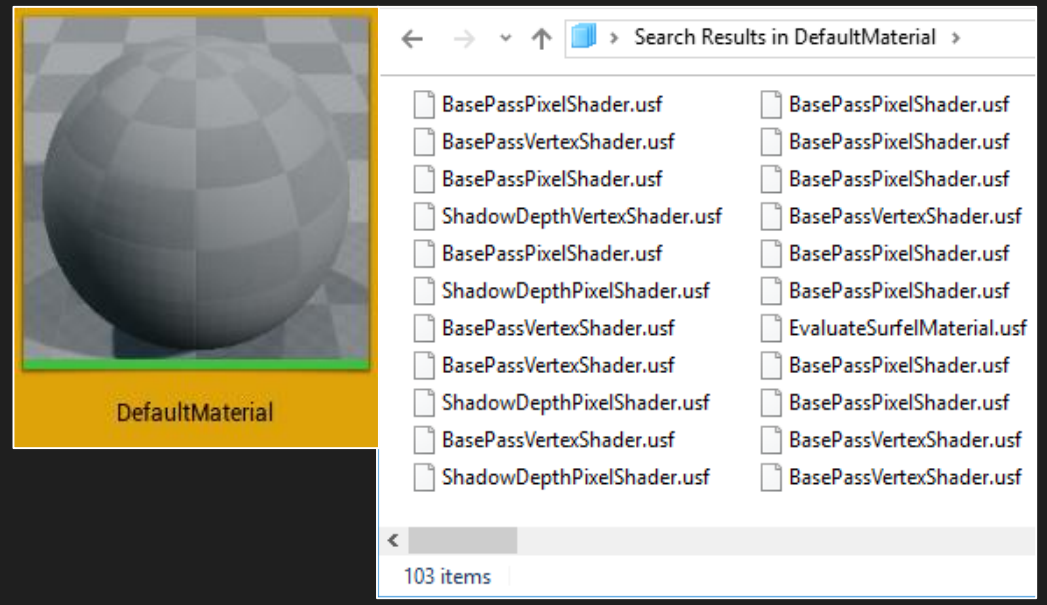
- 셰이더 개발
  - r.ShaderDevelopmentMode=1

- 셰이더 컴파일러에 들어가는 Preprocessed 셰이더 확인
  - r.DumpShaderDebugInfo=1
  - 프로젝트의 출력위치: %Project%/Saved/ShaderDebugInfo
  - 엔진의 출력위치: %Engine_root%/Saved/ShaderDebugInfo

- 위 정보가 패스가 너무 길다면...
  - r.DumpShaderDebugShortNames=1

- ASM을 확인하고 싶다면...
  - 셰이더 컴파일러가 바로 컴파일할 수 있도록 추가 파일도 출력
  - r.DumpShader.DebugWorkderCommandLine=1

※ 미리 세팅해놓고 싶다면, ConsoleVariables.ini에서 변경.



UE4 셰이더 디버깅

- 셰이더 컴파일러 디버깅
  - 셰이더 컴파일 워커(SCW) 사용 끄기 + 비동기 컴파일 끄기

BaseEngine.ini*

```
793
794  [DevOptions.Shaders]
795  ; See FShaderCompilingManager for documentation on what these do
796  bAllowCompilingThroughWorkers=False
797  bAllowAsynchronousShaderCompiling=False
```

※ 컴파일 내부 동작 확인

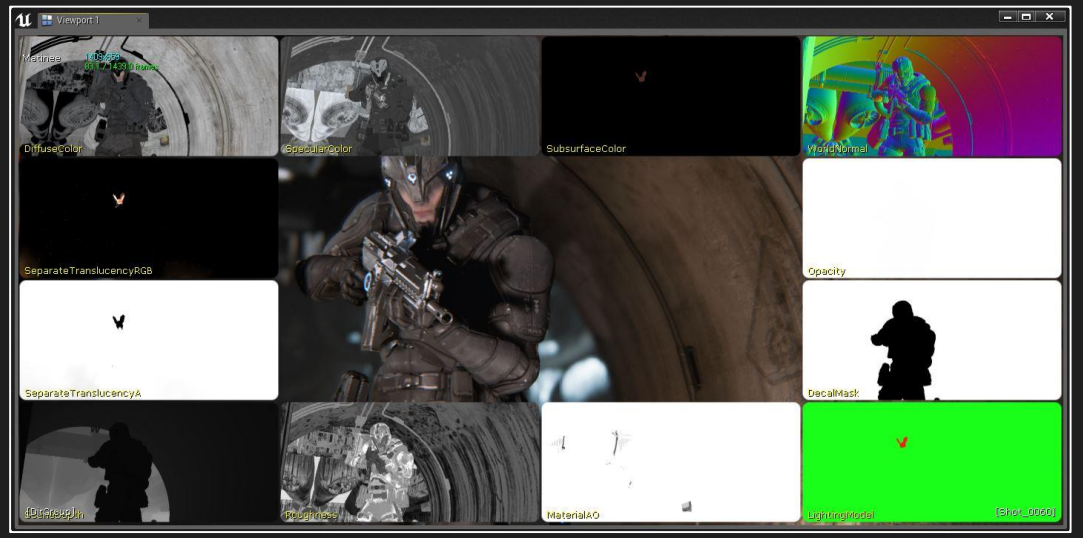
- D3D11 – CompileD3D11Shader()
- Metal – CompileShader_Metal()
- Vulkan – CompileShader_Windows_Vulkan()

UE4 셰이더 디버깅

- 셰이딩 연산 디버깅
  - ➔ 중간 데이터를 출력하도록 한 다음

```
OutColor = frac(WorldPosition / 1000);
```

  - ➔ VisualizeTexture나 Buffer Overview로 시각화



UE4 셰이더 디버깅

- 빠른 이터레이션
  - Global Shader: *recompileshaders changed* 활용
    - or Ctrl+Shift+. [period]
  - (Mesh) Material Shader: 머티리얼 에디터 활용

- 모든 UE4셰이더 재컴파일 시키기
  - common.ush 수정 후 엔진 재시작
  - 패키징된 프로젝트가 사용하는 모든 셰이더 확인
    - 타겟 플랫폼 설정 & r.DumpShaderDebugInfo=1

감사합니다.