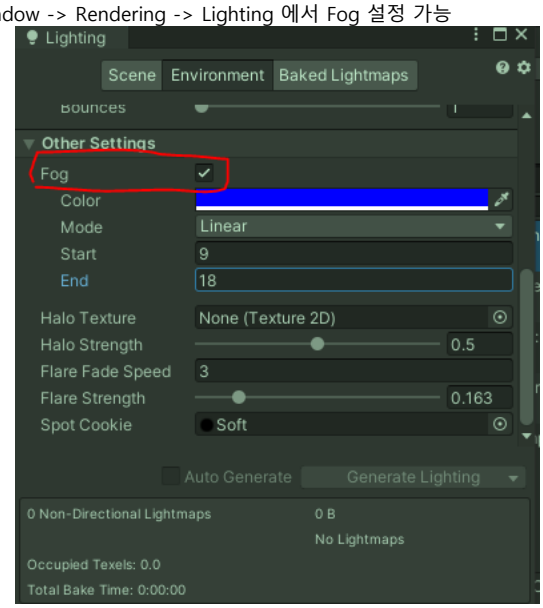
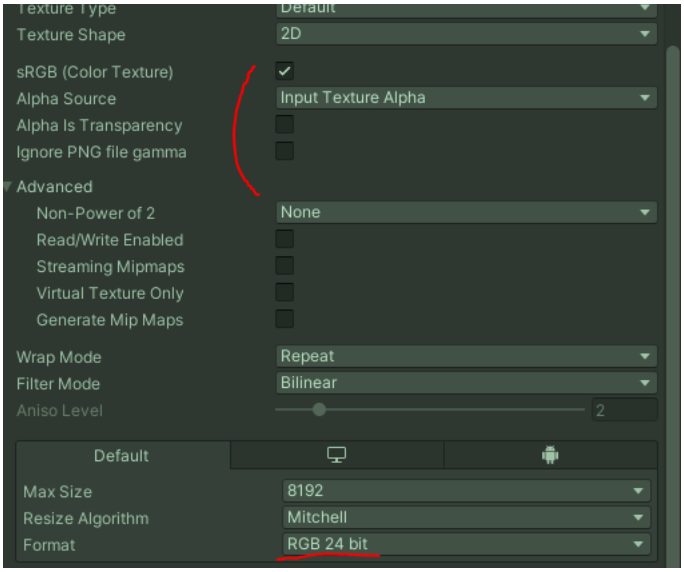
Fog 테스트

2022년 3월 1일 화요일 오후 1:38

- 계기
  - 씬전환하면서 맵들을 이동하는데 특정 맵에서
    스킬을 사용했을때 파티클 Effect 에 사용되는 Texture 들이 알파블렌딩이 제대로 안됐는지
    블렌딩되어 안보여야할 배경 부분이 자꾸 보이는거임.
  - 그래서 라이팅이나 shader texture 나 등등 보다가 , 카메라와의 거리에 따라서 이상하게
    해당 현상이 발생했다가 안했다가 하는걸 발견
  - 그러다가 아차 싶어서 Window -> Lighting -> Fog 쪽에 토글을 풀엇다가 해제해봣다가 했는데
    Fog 때문에 발생하던게 맞았음 .
  - 일단 결론으로 치면은 해당 스킬 파티클 effect 의 renderer 에서 사용하는 material 이
    Mobile/Particles/Additive 를 사용했는데 여기 코드에 보면 Fog 가 적용되게끔 하는
    세팅이 있어서 해당 texture 에 Fog 색상 연산까지 됐었고 Fog 색상연산은
    카메라와의 거리가 멀수록 Fog 에 설정된 Color 에 가깝게 되도록 lerp 연산이 들어가서
    거리가 멀어질수록 Texture 본래의 이미지는 사라지고 Rect 형태로 해당 Fog 색상으로
    가득 찬거처럼 보였던거였음
    - 해당 Shader 코드
      - https://gist.github.com/keijiro/c5b67a3e78201084e1e2
    - ```
      Fog { Color (0,0,0,0) }
      ```
      - 이 부분때문에 Fog 가 적용됐던거임 .
      - **Fog { Mode Off } 로 변경하면 Fog 적용 X**
- **주 테스트 목록**
  - Fog 연산 매크로 및 직접 구현으로 해보기
- **Fog 설정하기**
  - Window -> Rendering -> Lighting 에서 Fog 설정 가능
    - 

    - Linear 로 하면은 Fog 색상 변화를 Linear 하게 계산.
    - Linear 는 오히려 인간의 눈으로 봤을때 더 부자연스러워 보일수는 있음
      - 어두운 색상의 변화에 좀더 sensitive 하기 때문임.
      - 쉽게 말해 완전 칠흑같이 어두울때 [약간의 빛] 을 키는것과
        완전이 밝은 상태에서 같은 빛의 양인 [약간의 빛] 을 덜어내는 내는 것과
        인간의 눈을 전자를 좀 더 Sensitive 하게 느낀다는 거임.
        - 참고 - Gamma Space , Gamma Corrected, sRGB 등..
    - 그래도 일단 Linear 로 함.
- **리소스 가져오기**
  - 텍스쳐 가져오기
    - 

    - 위와 같이 Circle 형태의 텍스쳐 가져왔음
    - 테스트를 위해서 일단 Alpha Channel 은 필요없음 .
      ```



- 세팅은 이런식

○ Shader 생성

- Fog 지원 쉐이더 01 - BuiltIn Shader 쉐이더
  - https://gist.github.com/keijiro/c5b67a3e78201084e1e2
  - Mobile/Particles/Additive 쉐이더임
  - Shader 파일 하나 만들어서 위 코드를 해당 파일에 Copy & Paste 한 담에 Shader Path 만 마음대로 바꿔서 ㄱㄱ
  - `Cull Off Lighting Off ZWrite Off Fog { Color (0,0,0,0) }`
  - Fog 부분이 Fog 를 Enable 하는 Tag 임 .
- Fog 지원 쉐이더 02 - Custom Shader

```
Shader "FogTest/FogTest_Custom" {
	Properties{
		_MainTex("Particle Texture", 2D) = "white" {}
	}

		SubShader{
			Tags { "Queue" = "Transparent" "IgnoreProjector" = "True" "RenderType" = "Transparent" }
			Cull Off Lighting Off ZWrite Off
			// == 알파 블렌딩 활성화 ==
			// Src 즉 Drawing 하려는 Transparent 오브젝트의 Color * 해당 Vertex 의 Alpha + 이미 Drawing 된 오브젝트 즉 Dest 의 Colo
			Blend SrcAlpha One

			 Pass
			{
				CGPROGRAM
				#pragma vertex vert
				#pragma fragment frag
				#pragma multi_compile_fog

				#include "UnityCG.cginc"

					struct appdata
					{
						float4 vertex : POSITION;
						float2 uv : TEXCOORD0;
						float4 color : COLOR;
					};

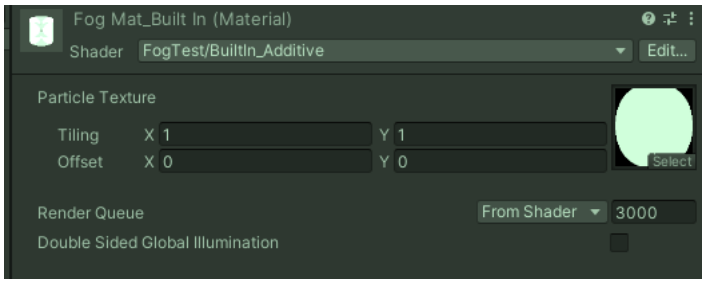
					struct v2f
					{
						float2 uv : TEXCOORD0;
						// 해당 Fragment 의 Camera 기준으로 가까울수록 1 , 멀수록 0 에 가까운 float1 변수 선언
						// 이 변수로 해당 Fragment 에 Fog 색상을 얼마나 적용시킬지 정할 수 있음.
						// 왜냐면 멀수록 0 이니까
						// 즉 이 값은 Fog 의 Factor
						UNITY_FOG_COORDS(1)
						float4 vertex : SV_POSITION;
						float4 color : COLOR;
					};

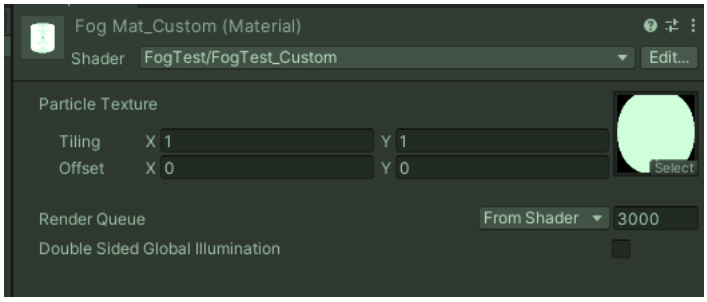
					sampler2D _MainTex;
					float4 _MainTex_ST;

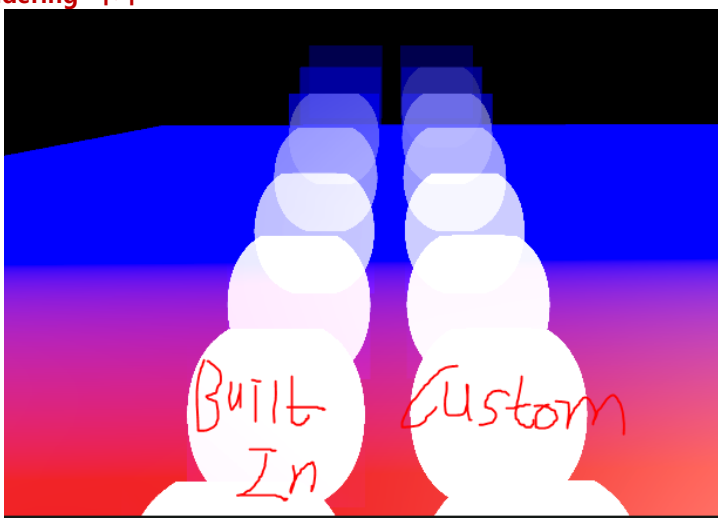
					v2f vert(appdata v)
					{
						v2f o;
						o.vertex = UnityObjectToClipPos(v.vertex);
						o.uv = TRANSFORM_TEX(v.uv, _MainTex);
						o.color = v.color;
						// 이 Macro 는 해당 Vertex 의 z 값
						UNITY_TRANSFER_FOG(o,o.vertex);
						return o;
					}

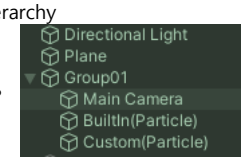
					float4 frag(v2f i) : SV_Target
```

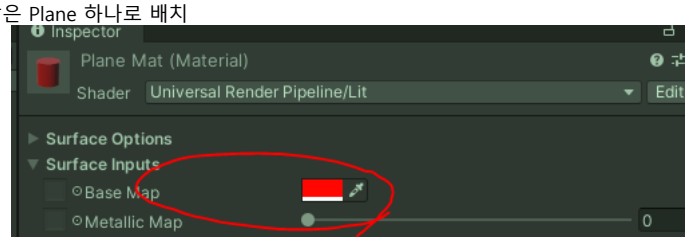
```
                {
                    // 여기서 Color 는 테스트할때 ParticleSystem Component 에서 넣게끔 설정할 예정
                    // 즉 거기서 Alpha 값도 설정할 것임
                    float4 col = tex2D(_MainTex, i.uv) * i.color;
                    // 이 Macro 는 Fragment 의 Position 값에 의해 계산된 Fog 의 Factor 값이 들어있는
                    // fogCoord 를 넘겨주고 Fog 가 적용되지 않은 Original 형태의 Color 값인 col 을 넘겨주면
                    // col 에다가 Fog 가 적용된 색상 값을 넣어주는 Macro 임
                    // 걍 쉽게 말하면 , Fog 색상 적용해서 Col 에 넣는다
                    UNITY_APPLY_FOG(i.fogCoord, col);
                    return col;
                }
                ENDCG
            }
        }
    }
```

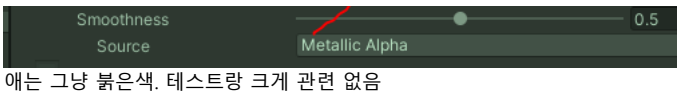
- Shader 파일 하나 더 만들어서 위 코드 Copy & Paste
- Material 만들기
  - 같은 Texture 를 사용하게끔 해서 두개 생성 ㄱㄱ
    - FogMat_BuiltIn
      FogMat_Custom
    - 

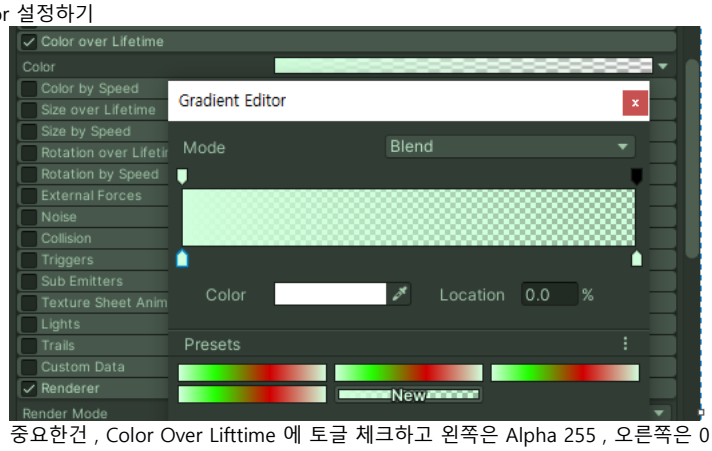
    - 

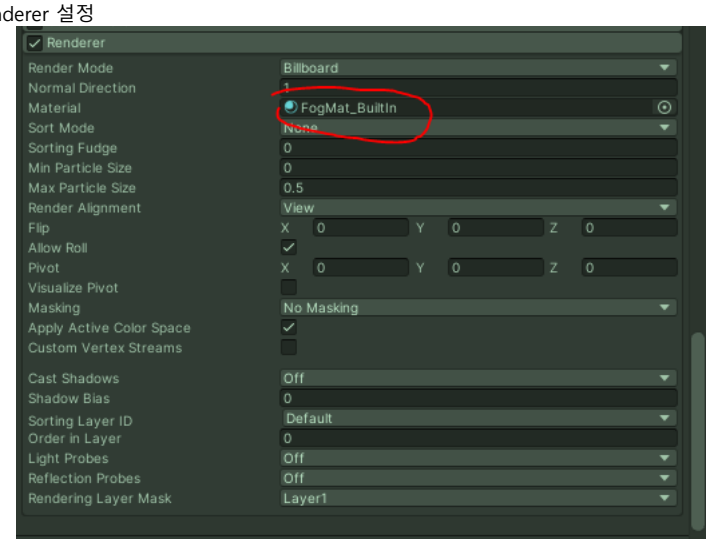
- **Rendering 하기**
  - ○ 

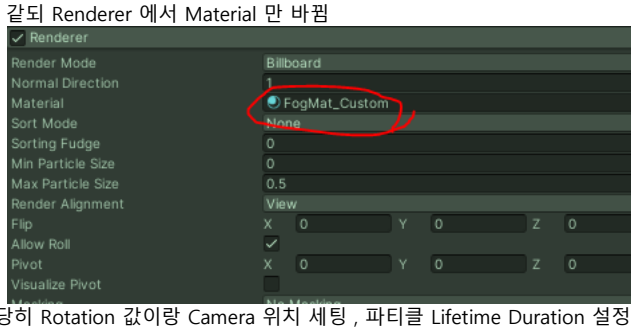
    - 이렇게 만들 예정. 왼쪽은 BuiltIn Shader 를 사용하는 Material 로 렌더하고 오른쪽은 Custom 사용하게끔 설정할 예정
  - Hierarchy
    - 

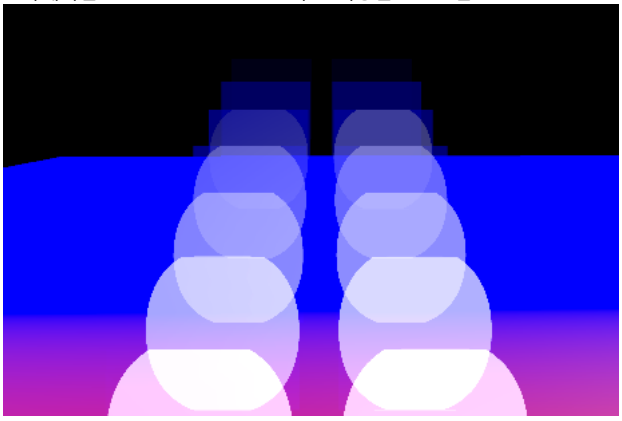
    - 이런 모양
  - 바닥은 Plane 하나로 배치
    - 

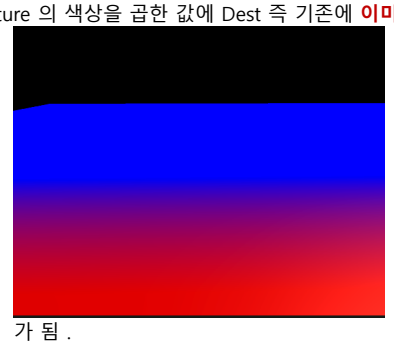
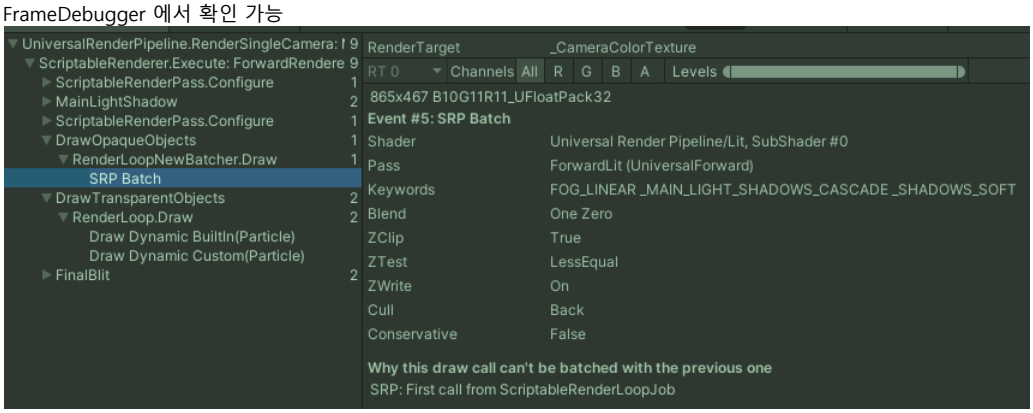


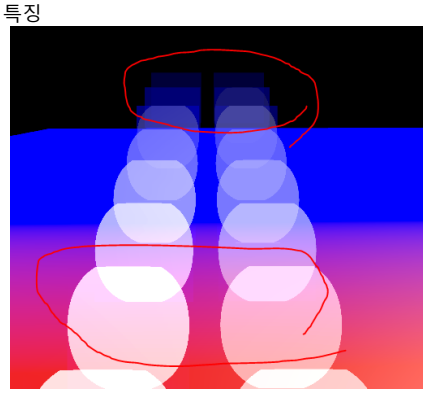
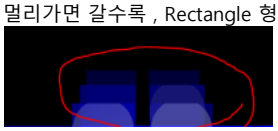
- 애는 그냥 붉은색. 테스트랑 크게 관련 없음
  - Render 하기 위해서 ParticleSystem 사용
    - Built In Shader 로 렌더링할 Particle 세팅
      - Color 설정하기
        - 

        - 중요한건 , Color Over Lifttime 에 토글 체크하고 왼쪽은 Alpha 255 , 오른쪽은 0 으로 해서 앞으로 가면서 투명되게끔 ㄱㄱ
        - **여기서 설정한 값이 Color 값이 Shader 에 COLOR 값으로 전달됨**
      - Renderer 설정
        - 

        - Material 설정하면 끝
      - 나머지는 .. 그냥 적당히 화면에 이미지처럼 날아가면서 보이게끔 설정 ㄱㄱ
    - Custom Shader 로 렌더링할 Particle 세팅
      - 위와 같되 Renderer 에서 Material 만 바꿈
        - 

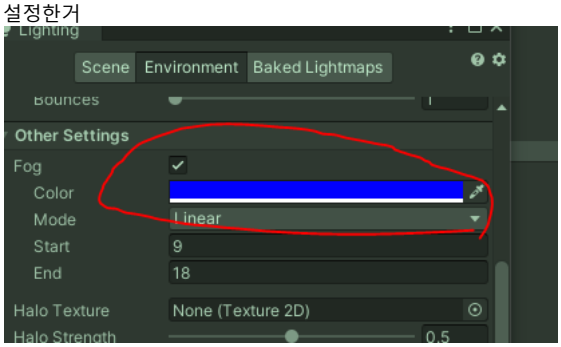
    - 이렇게 하고 적당히 Rotation 값이랑 Camera 위치 세팅 , 파티클 Lifetime Duration 설정 등 하면 됨 .
- **중간 점검**
  - 자 이렇게까지 하고 플레이 해보면 , 대략 이런식임 .
  - 참고로 카메라는 Solid Color 로 Clear 하고 색상은 Black 임
    - 

- 자 보면은 Fog 색상이 Blue 이기 때문에 Camera 에서 멀수록 Blue 가 됨 .
  반면에 , 가까우면 가까울수록 바닥 색상인 Red 에 가까워짐.
- **여기서 특징 정리**
  - 알파 블렌딩
    - 알파 블렌딩은 SrcAlpha One 이기 때문에 Particle 에서 지정해준 알파 1 -> 0 에
      Texture 의 색상을 곱한 값에 Dest 즉 기존에 **이미 그려진 색상** 즉 이 테스트에서는
      - 
      - 가 됨 .
      - 이건 FrameDebugger 에서 확인 가능
        - 

        - 보면은 불투명 오브젝트 즉 Opaque 들 그린 다음 Transparent Object 를 그림
        - 여기서 Transparent 오브젝트들이 파티클 오브젝트임
  - Fog 특징
    - 
    - Fog 가 적용된 Shader 에서는 Camera 에 가까울수록 Fog 가 적용되지 않은 Color 로 적용되고
      멀수록 Fog Color 에 가까운 색상으로 적용됨.
    - 그래서 , 가까이 있을수록 원래 색상이 그대로 Rendering 이 됨 .
    - 여기서는 AlphaBlending 이 SrcAlpha One 이기 때문에 배경색같이 Black 인 Texture 를 사용할때
      0 + Dest Color 가 됨.
      - 즉 이미지 처럼 카메라랑 가까울수록 배경이 Transparent 로 보임 . 배경색으로 블렌딩됐기 때문에
      - 즉 Fog 적용이 약한 상태
    - 근데 멀리가면 갈수록 , Rectangle 형태의 Fog 색상과 배경색이 blending 되어 나타나게됨.
      - 
    - 이건 왜그러냐 ?
      - 왜냐면은 , 기존에 Fog 가 약하게 적용될때는 Fragment Shader 에서의 Return 값이
        Texture Color 와 Vertex Color 만 적용이 됐었는데 이제는 그 Color 값에
        Fog 연산이 들어가게 되면서 Fog 색상 그 자체가 되어버리는 거임.
      - 즉 하얀색에도 Blue 가 섞이고 검은색에도 Blue 가 섞인거임 .
      - **이 연산은 내부적으로 lerp 로 계산하기 때문에 단순히 그냥 멀면 멀수록 Blue 가 되어 버림.**
    - 정리하면 Fog 가 적용이 안될수록 원래 색상 그대로 사용했기 때문에 검은 색상은 배경에 그대로 + 가 되어
      Transparent 화가 된 반면 , 카메라와 멀어서 Fog 가 적용이 됐을때는 Fog Color 에 lerp 연산이 되어
      기존 색상이 Blue 에 가까워진 것임 .
      - 즉 잘렸던 Black 배경 색상이 Blue 가 되면서 마치 AlphaBlending 자체에 문제가 있는거 처럼 보였던거임 .
      - 그래서 결국은 Fog 가 적용되면 될수록 네모가 나오는거 (Clip 된게 아니라 배경색과 블렌딩되어 Clip 된거처럼 보인거)
      - 만약 이렇게 Fog 가 적용되어도 Rectangle 로 나오는게 아니라, 아예 하얀 Circle 만 나오게 하려면
        texture 에 AlphaChannel 을 할당해서 값을 넣으면됨 .

- **Fog 직접 구현**
  - Fog 직접 구현이라고 하기엔 뭐하고 , 그냥 Unity 가 내부적으로 Fog 색상 연산을 어떻게 하는지 그걸 보고 직접 테스트 ㄱㄱ
  - 힌트는 이 녀석
    - `UNITY_APPLY_FOG(i.fogCoord, col);`
    - 애가 어떻게 Camera 기준으로 해당 Fragment 의 Normalize 된 위치 z 값 (0 ~ 1) 을 저기에 넣고 본래의 color 를 넣으면 col 값에 그 result 를 넣는지를 봐야함 .
    - 해당 Macro 는 UnityCG.inc 에 위치
      - 참고
        - https://github.com/TwoTailsGames/Unity-Built-in-Shaders/blob/master/CGIncludes/UnityCG.cginc
  - *UNITY_APPLY_FOG* 매크로
    - 위 주소에서 UNITY_APPLY_FOG 을 찾으면

```
#ifdef UNITY_PASS_FORWARDADD
    #define UNITY_APPLY_FOG(coord,col) UNITY_APPLY_FOG_COLOR(coord,col,fixed4(0,0,0,0))
#else
    #define UNITY_APPLY_FOG(coord,col) UNITY_APPLY_FOG_COLOR(coord,col,unity_FogColor)
#endif
```

    - 이렇게 나옴 . 그리고 아마 밑에있는 Macro 가 실행이 될거임. unity_FogColor 는 전역적으로 접근 가능한 FogColor 임
      - 전에 설정한거
      - 

      - 이거
    - 무튼 UNITY_APPLY_FOG_COLOR 에다가 해당 인자들을 넘김
  - *UNITY_APPLY_FOG_COLOR* 을 내부적으로 사용하기 때문에 , 이걸 또 검색 ㄱ

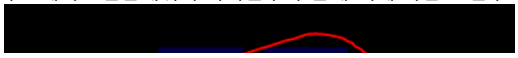
```
#if defined(FOG_LINEAR) || defined(FOG_EXP) || defined(FOG_EXP2)
    #if (SHADER_TARGET < 30) || defined(SHADER_API_MOBILE)
        // mobile or SM2.0: fog factor was already calculated per-vertex, so just lerp the c
        #define UNITY_APPLY_FOG_COLOR(coord,col,fogCol) UNITY_FOG_LERP_COLOR(col,fogCol,(coo
    #else
        // SM3.0 and PC/console: calculate fog factor and lerp fog color
        #define UNITY_APPLY_FOG_COLOR(coord,col,fogCol) UNITY_CALC_FOG_FACTOR((coord).x); UN
    #endif
```

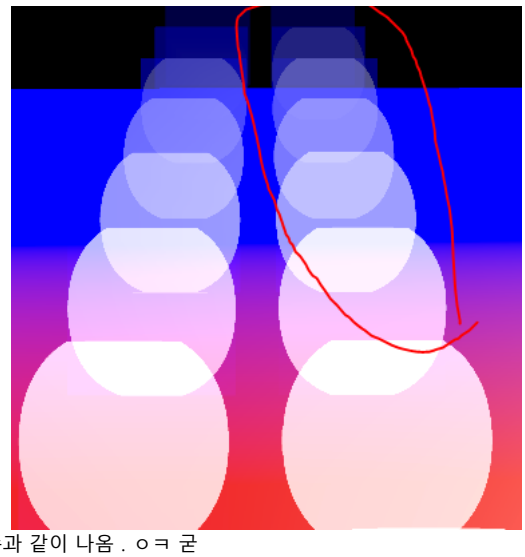
    - 자 이렇게 있음 .
    - 10000% 확실치는 않지만 테스트해봤을때는 위에 것이 호출되는거같음. 밑에는 모르겟지만 비슷하겠지 ??
  - 내부적으로 UNITY_FOG_LERP_COLOR 를 쓰기 때문에 또 찾아보자 .
  - *UNIT_FOG_LERP_COLOR*

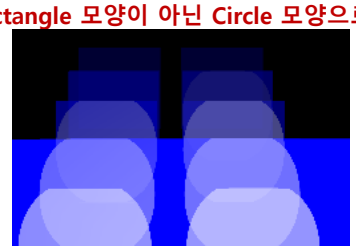
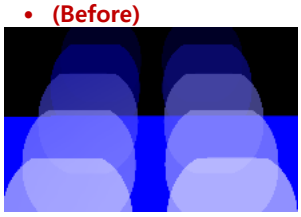
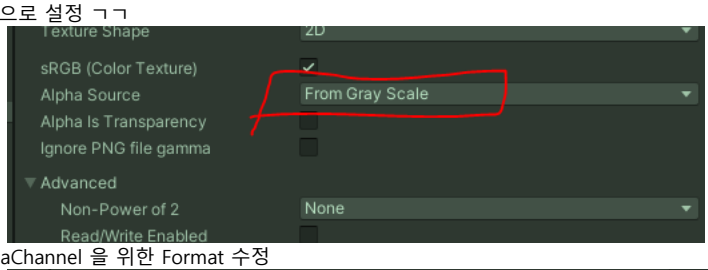
```
#define UNITY_FOG_LERP_COLOR(col,fogCol,fogFac) col.rgb = lerp((fogCol).rgb, (col).rgb, satu
```

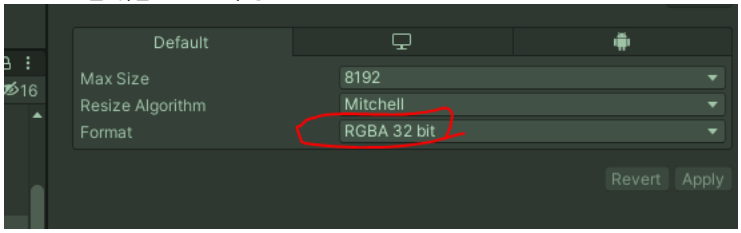
    - 이 매크로에서 최종적인 계산 식이 발견됨 . ㅇㅇ
      - lerp(a, b, f) 인데 여기서 a 는 srcColor , b 는 dstColor , t 는 factor 임 . ㅇㅋ ? 그냥 보통 쓰는 lerp임 . CG 랑 HLSL 에 있는 함수임
      - saturate 는 이름이 개그지같은데 , 그냥 clamp 임
        - 0 ~ 1 을 벗어나는 값들을 clamp 해주는 함수임 .
          - e.g.
            - -3.5 => 0
            - 2.1 => 1
            - 0.5 => 0.5
    - 저 식을 적용시키면

```
float4 frag(v2f i) : SV_Target
{
    float4 col = tex2D(_MainTex, i.uv);
    float4 result = lerp(col, unity_FogColor, saturate(1 - i.fogCoord.x));
    result *= i.color;
    return result;
}
```

    - 이렇게 된다
    - i.fogCoord.x 카메라에 가까울수록 1 , 멀수록 0 인 값이 들어가는듯함
    - 그래서 1 에서 - 연산해줘서 가까울수록 본래 색에 가깝고 멀수록 fog 색에 가깝게 연산함 .
    - 

- 
  - 기존과 같이 나옴 . ㅇㅋ 굳
- 정리하자면 , Fog 를 적용했을때, 기존에 Fog 없이 계산된 Color 에다가 Fog Color 로 Camera 와의 거리따라 lerp 처리가 되므로 , 파란색으로 되는거고. 그 뒤에 기존과 같이 배경색과 blending 이 되는거임 .

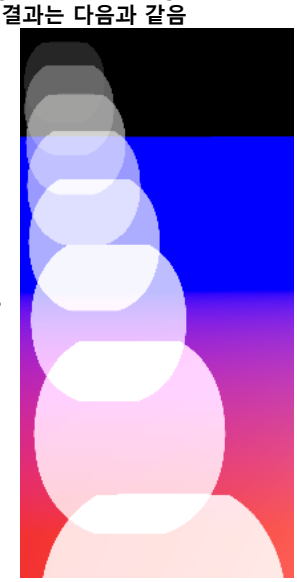
- **Rectangle 모양이 아닌 Circle 모양으로 나오게 하기**
  - 
    - **(Before)**
  - 
    - **(After)**
  - **Texture 에 AlphaChannel 추가**
    - 현재 Texture 에 AlphaChannel 을 추가해서 아예 배경을 Alpha 0 으로 설정해서 SrcColor 가 0 안나오게 하면 됨.
    - Alpha 가 0 이 되면 , Fog 에서 Color 가 Fog Color 가 됐다 하더라도 , Shader 에서 OM (Output merger) stage 에서 Blending 할때 지정한 Blending Function 인 SrcAlpha 에서 SrcColor * 0 이 되므로, 잘리게됨.
    - Alpha Channel 을 위한 Setting 하기
      - 해당 Png 파일에 AlphaChannel 이 없으므로, Gray Scale 로 해서 하얀색에 가까울수록 Alpha 가 1 , 검은색에 가까울수록 0 인 방식으로 설정 ㄱㄱ
      - 

      - AlphaChannel 을 위한 Format 수정
      - 

    - 이제 스크립트로 가서 Texture 의 Alpha 값을 Fragment Shader 의 return Color 의 alpha 값에 넣어줘야함

```
float4 frag(v2f i) : SV_Target
{
    float4 col = tex2D(_MainTex, i.uv) * i.color;
    float4 result = lerp(col, unity_FogColor, saturate(1 - i.fogCoord.x));
    result.a = col.a;
    return result;
}
```

    - 알파 넣은거 끝임.
  - **Fog 끄기**

- **Fog 를 끈다는건 Fog 색상 연산을 안한다는 것임 .**
  즉 결과는 다음과 같음
  - 
- *BuiltIn Shader*
  - *Fog { Color(0,0,0,0) }* 을 다음과 같이 수정
    Fog { Mode Off }
- *CustomShader*
  - Fog 끄는거는 그냥 Fog 연산은 안하면 됨.
  - 즉 Texture Color * Vertex Color 만 연산 ㄱㄱ

```
float4 frag(v2f i) : SV_Target
{
    return tex2D(_MainTex, i.uv) * i.color;
}
```