



고해상도 렌더 파이프라인의 조명 및 환경

(Unity 6 에디션)



목차

들어가는 말	11
이 가이드의 사용법	12
HDRP 조명 및 환경	13
시간의 유형	14
Unity 6.1의 HDRP 업데이트	17
조명	17
환경 효과	19
성능 및 최적화	20
패키지 관리자 샘플	21
렌즈 플레어	22
모발	23
HDRP 설치	24
템플릿을 사용하여 HDRP 프로젝트 만들기	25
수동으로 HDRP 설치	25
HDRP 샘플 씬	26
더 많은 HDRP 샘플 콘텐츠	28
프로젝트 설정	31
그래픽스 설정	32
품질 설정	33
HDRP 최적화	35
HDRP 전역 설정	35
HDRP 기능 활성화	36
볼륨	37
로컬 볼륨과 글로벌 볼륨	38
성능 팁	40

볼륨 프로파일	40
볼륨 오버라이드	41
오버라이드 워크플로	42
블렌딩 및 우선순위.....	43
노출	45
노출 값의 이해	45
노출 값 수식	47
노출 오버라이드	48
Fixed 모드	48
Automatic 모드.....	48
Metering Mode 옵션.....	49
Automatic Histogram	50
Curve Mapping.....	51
물리적 카메라	52
더 많은 물리적 카메라 파라미터	53
광원	55
광원 유형	55
셰이프	56
색상 및 온도	57
더 많은 프로퍼티.....	58
광원 레이어.....	59
광원 앵커.....	60
물리 광원 단위 및 강도	62
단위	62
일반적인 조명 및 노출 값	63
IES 프로파일 및 쿠키	64

HDRP 전역 조명	66
HDRP 전역 조명 기능	67
베이킹된 전역 조명	68
라이트매핑 워크플로	69
라이트맵 최적화	71
GPU 라이트매핑	72
라이트맵 UV	72
조명 베이킹 인터랙티브 프리뷰(Unity 6.1)	74
다이내믹 전역 조명	74
SSGI(스크린 공간 전역 조명)	75
트레이싱 모드	76
Mixed 트레이싱 모드	77
RTGI(레이트레이싱 전역 조명)	77
Enlighten Realtime GI 지원 중단	78
라이트 프로브와 적응적 프로브 볼륨	79
라이트 프로브와 SSGI/RTGI 비교	80
라이트 프로브 그룹	80
런타임에 라이트 프로브 이동	81
적응적 프로브 볼륨	82
스폰자 궁전이란?	84
적응적 프로브 볼륨 사용	84
조명 시나리오 에셋	88
조명 시나리오 관리자	91
APV 문제 해결	91
유효하지 않은 프로브 해결하기	92
빛 번짐 현상 해결하기	94
Probe Adjustment Volume 사용	95

경계 해결하기	96
렌더링 레이어 마스크	98
APV의 스카이 오클루전	98
스카이 오클루전 활성화	99
스카이 오클루전 디버깅	99
하늘 방향	101
APV 데이터 로딩 최적화.....	101
APV 데이터 스트리밍	101
AssetBundles 또는 Addressables에서 APV 데이터 로딩	102
런타임에 라이트 프로브 베이킹	102
그림자.....	103
새도우 맵.....	103
그림자 캐스케이드	104
컨택트 새도우	106
마이크로 새도우	108
면 광원 소프트 새도우	109
반사	110
스크린 공간 반사.....	111
반사 프로브.....	112
최적화 팁.....	114
Planar Reflection Probe 컴포넌트	114
하늘 반사.....	115
반사 계층 구조	116
프록시 볼륨	117
실시간 조명 효과.....	118
스크린 공간 앰비언트 오클루전	118
스크린 공간 굴절.....	120

광원 렌더링 레이어	121
렌더링 레이어 16개와 32개 비교	121
렌더링 레이어 설정	121
광원에 렌더링 레이어 사용	122
그림자용 렌더링 레이어 사용	123
레이트레이싱 및 패스트레이싱	125
Unity 6의 새로운 기능	125
설정	126
오버라이드	127
인라인 레이트레이싱 지원(PC 및 콘솔)	133
레이트레이싱 파이프라인	133
인라인 레이트레이싱	134
성능	134
패스트레이싱	136
DirectX 12	138
Unity 6의 샘플	139
환경 조명	140
기본 조명 데이터 에셋	140
비주얼 환경	142
HDRI Sky	143
HDRI Sky 애니메이션	144
Gradient Sky	144
Physically Based Sky	144
색상 팁	146
Unity 6의 PBR 하늘 업데이트	146
대기 산란	148
렌더링 공간	149

Environment 샘플	149
터레인.....	151
터레인 생성	152
조형	152
텍스처링 및 디테일링	153
나무와 초목	154
SpeedTree 통합.....	154
Terrain Tools 패키지.....	155
터레인 페인팅	156
Noise Editor	159
Terrain Toolbox	159
터레인용 레이트레이싱 지원	160
HDRP Terrain 데모	161
구름	163
구름 레이어	164
대기 및 태양 기반 조명	165
볼류메트릭 구름	166
Unity 6의 업데이트	169
Environment 샘플	169
안개 및 대기 산란	173
글로벌 안개	173
볼류메트릭 조명	176
볼류메트릭 조명 및 그림자 팁	177
Local Volumetric Fog	179
HDRP 물 시스템.....	181
Unity 6.1의 새로운 기능	182
물 시스템 시작하기.....	183

Water Surface 컴포넌트	185
물 데칼(Unity 6.1)	186
Unity 6의 Water 샘플	187
빙하	187
섬	190
풀	191
비	192
동굴	192
밀려오는 파도	194
수중	196
셰이더 및 머티리얼	199
머티리얼 샘플	200
머티리얼 배리언트	201
머티리얼 프로퍼티	202
HDRP 마스터 스택	204
Material Type 프로퍼티	207
볼류메트릭 셰이더 그래프 안개	208
전체 화면 셰이더 그래프	209
투명도	211
Lit Material 샘플	211
Transparent Material 샘플	212
Shader Graph 노드	213
Compute Thickness	214
피하 산란 및 반투명	215
SpeedTree 및 Transmission Mask	218
Unity 6의 새로운 기능: 향상된 피부 렌더링	218

데칼	220
셰이더: 모피와 모발	221
Unity 6.1의 새로운 기능:.....	222
포워드 렌더링 및 디퍼드 렌더링 비교	223
렌더링 경로 커스터마이징	224
렌더링 경로에 대한 자세한 내용	225
포워드 렌더링	225
디퍼드 셰이딩	226
안티앨리어싱.....	227
MSAA(멀티 샘플 안티앨리어싱).....	227
포스트 프로세싱 안티앨리어싱.....	229
포스트 프로세싱	231
포스트 프로세싱 오버라이드	233
그림자, 미드톤, 하이라이트	235
블룸	235
덱스오브필드(피사계심도)	236
화이트 밸런스	238
색상 커브.....	239
색상 조정.....	239
채널 믹서.....	239
렌즈 왜곡.....	240
비네팅	241
모션 블러.....	241
렌즈 플레어.....	242
Lens Flare (SRP) 컴포넌트	243
스크린 공간 렌즈 플레어.....	243

시작하기	245
다이나믹 해상도	247
Catmull-Rom	248
CAS	248
NVIDIA DLSS(NVIDIA RTX GPU 및 Windows용)	249
AMD FSR(크로스 플랫폼)	249
TAA Upscale(크로스 플랫폼)	250
시공간 포스트 프로세싱(크로스 플랫폼)	251
렌더링 디버거	252
색상 모니터	255
Runtime Frame Stats	256
HDR 디스플레이 지원	257
VFX Graph 지원	259
커스텀 HLSL 블록과 오퍼레이터	259
볼류메트릭 안개 출력	260
레이트레이싱 지원	261
최적화	262
스크립터블 렌더 파이프라인 배치	262
BatchRendererGroup	262
GPU 상주 드로어	263
GPU 오클루전 컬링	264
DX12에서 에디터를 통한 그래픽스 잡 지원	265
Variable Rate Shading (Unity 6.1)	266
다음 단계	267
더 많은 자료	267

들어가는 말

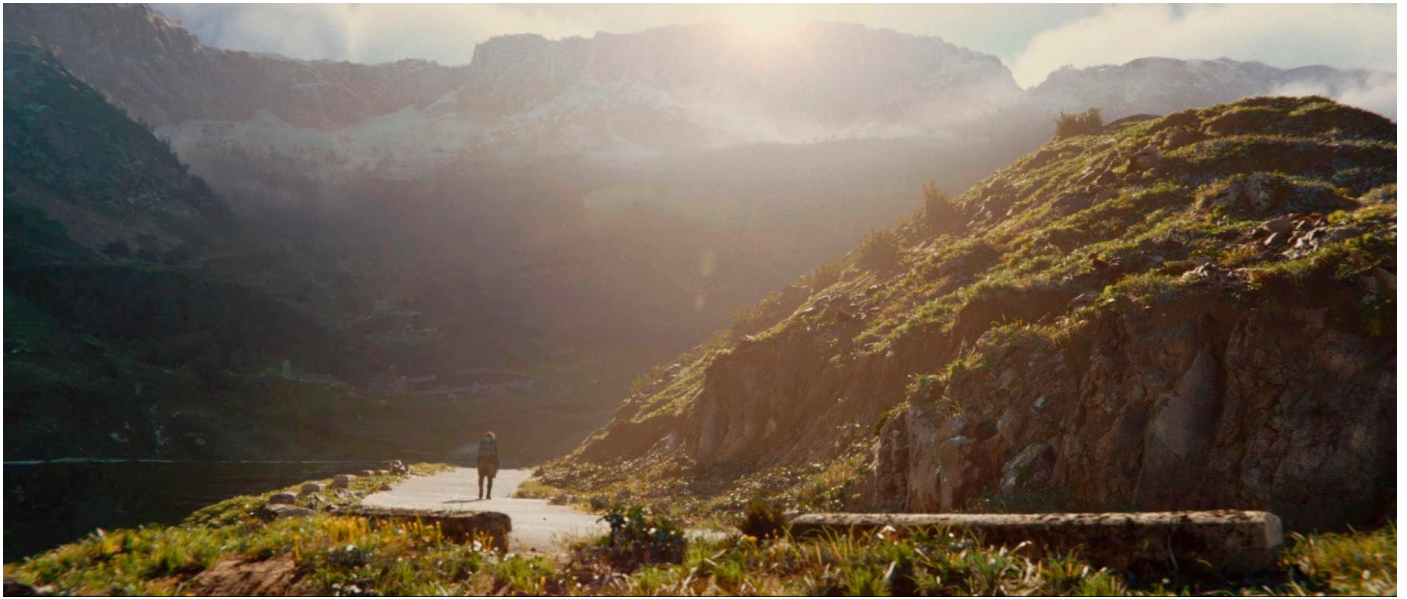
게임 월드는 단순한 배경 이상의 역할을 합니다. 게임의 모든 순간들이 펼쳐지는 인터랙티브한 무대이죠.

Unity 6에서 HDRP(고해상도 렌더 파이프라인)로 월드를 제작하면 실시간 그래픽스의 한계를 넓히는 근사한 환경을 만들 수 있습니다. HDRP는 물리적으로 정확한 조명, 높은 충실도를 보여 주는 머티리얼과 시네마틱 효과로 씬을 한층 더 몰입감 넘치게 만들어 줍니다.

불류메트릭 안개와 그림자로 씬에 깊이감을 더하세요. SpeedTree의 다이내믹 초목으로 랜드스케이프에 생동감을 부여하고, 데칼 프로젝터로 표면에 먼지, 틈, 마모를 적용하여 세월의 흔적을 자연스럽게 표현하세요. 브러시드 메탈의 은은한 광택부터 레이어링된 유리를 통과하는 빛의 복잡한 움직임까지, 레이트레이싱으로 새로운 차원의 사실감을 구현하세요.

이 HDRP 가이드를 업데이트하여 유니티가 제공하는 최신 월드 제작 툴에 대한 내용을 담았습니다. Terrain Tools를 사용하여 산을 옮기거나 협곡을 조형해 보세요. 클라우드 레이어로 하늘에 구름을 더하고, 조명 시나리오를 사용하여 낮에서 밤으로 실시간으로 전환하세요. HDRP의 고급 물 시스템을 사용하면 바다와 강을 채우고 파도, 커스틱, 거품까지 생생하게 구현할 수 있습니다.

여러분의 월드에는 어떤 스토리가 있나요? HDRP로 비전을 실현해 보세요.



HDRP로 사실적인 게임 월드를 제작할 수 있습니다. 위의 두 이미지는 Unity의 하이엔드 데모 [시간의 유형](#)에 등장하는 장면입니다.

이 가이드의 사용법

본 전자책은 HDRP의 다양한 툴을 쉽게 찾아볼 수 있도록 구성된 모듈식 참조 가이드입니다. 조명, 환경 효과, 렌더링 기법, 최적화 전략 등 원하는 주제로 곧바로 이동할 수 있습니다.

각 챕터는 독립적으로 구성되어 있지만, 여러 주제의 내용은 서로 관련되어 있습니다. 예를 들어, HDRP의 조명 시스템은 물리 광원 단위, 볼륨, 포스트 프로세싱 기법과 같은 여러 섹션에 걸쳐 설명되어 있습니다.

마찬가지로, 터레인, 물, 하늘, 안개 시스템과 같은 환경 관련 기능은 함께 작동하여 몰입감 넘치는 월드를 구현합니다. 이 가이드북을 효과적으로 활용하려면 HDRP의 기능을 종합적으로 이해할 수 있도록 관련 챕터를 함께 읽어 보세요.

HDRP 조명 및 환경

HDRP는 씬을 현실 조명과 흡사하게 렌더링할 수 있도록 다음과 같은 다양한 기능으로 Unity의 기존 조명 시스템을 확장합니다.

- **물리 광원 단위 및 고급 조명:** HDRP는 현실에서 사용하는 조명 강도와 단위를 사용합니다. 실제 광원과 사양을 동일하게 맞추고 물리적 카메라를 사용하여 노출을 설정할 수 있습니다. 다양한 스폿 광원 및 면 광원 셰이프 옵션으로 광원 배치를 제어할 수 있습니다. 스크린 공간 전역 조명, 스크린 공간 굴절 등 실시간 효과를 적용할 수 있습니다.
- **하늘 풍경:** 다양한 기술로 자연스러운 하늘을 생성해 보세요. Physically Based Sky 볼륨 오버라이드를 사용하여 대기를 점차적으로 시뮬레이션하거나, 볼류메트릭 구름 또는 구름 레이어를 추가하거나, HDRI를 적용하여 정적 하늘을 시뮬레이션할 수 있습니다.
- **터레인:** HDRP로 강화된 조형 툴, 텍스처 레이어링, 커스텀 브러시를 사용하여 사실적인 랜드스케이프를 제작해 보세요. 하이트맵으로 지형을 칠하고 바람에 흔들리는 초목을 추가하여 풍성하고 역동적인 씬을 만들 수 있습니다.
- **물 시스템:** HDRP의 물 시스템은 상호 작용 가능한 수면을 시뮬레이션합니다. 물리적으로 정확한 파도 시뮬레이션, 물 흐름, 거품을 추가하여 현실적인 씬을 만들어 보세요.
- **안개:** 안개로 씬에 깊이와 입체감을 더해 보세요. 볼류메트릭 효과를 활성화하여 안개 효과를 전경 오브젝트와 통합하고 영화 같은 빗줄기를 렌더링할 수 있습니다. 또한 볼류메트릭 조명과 그림자를 광원별로 제어하고, Local Volumetric Fog 컴포넌트를 사용해 3D 마스크 텍스처로 안개 밀도를 미세하게 조정할 수 있습니다.
- **볼륨 시스템:** HDRP에는 카메라 위치나 우선순위에 따라 다양한 조명 효과와 설정을 활용할 수 있는 직관적인 시스템이 있습니다. 볼륨을 레이어링하고 블렌딩하여 씬의 모든 공간을 전문가 수준으로 제어해 보세요.

- **포스트 프로세싱:** HDRP 포스트 프로세싱은 기존 볼륨 시스템에 여러 볼륨 오버라이드를 얹는 방식으로 제어합니다. 안티앨리어싱, 톤 매핑, 컬러 그레이딩, 블룸, 덤스오브필드뿐만 아니라 여러 효과를 추가할 수 있습니다.
- **고급 그림자 옵션:** HDRP는 사실적인 그림자를 만들고 성능을 제어할 수 있는 고급 옵션을 제공합니다. 틸트, 필터링, 해상도, 메모리 할당량, 업데이트 모드를 수정할 수 있습니다. 컨택트 새도우와 마이크로 새도우로 작은 디테일을 살리고 더 깊이 있는 씬을 연출하세요.
- **고급 반사 옵션:** 여러 기술을 사용하여 반사 표면을 렌더링할 수 있습니다. 반사 프로브는 기존 반사 매핑 접근 방식을 제공하며, 평면 반사 프로브는 평평한 표면에 더 다양한 고급 옵션을 제공합니다. SSR(스크린 공간 반사)은 덤스 버퍼를 사용하여 실시간 기술을 제공합니다.
- **확장성:** HDRP는 Unity의 [스크립터블 렌더 파이프라인](#)을 기반으로 합니다. 숙련된 테크니컬 아티스트나 그래픽스 프로그래머라면 기본 제공되는 기능 이상으로 파이프라인을 확장할 수 있습니다.

HDRP를 처음 사용해 본다면 [고해상도 렌더 파이프라인 시작하기 튜토리얼](#)을 확인해 보세요.

시간의 유령

시간의 유령(Time Ghost)은 더 블랙스미스(The Blacksmith), 아담(Adam), 사자의 서(Book of the Dead), 더 헤레틱(The Heretic), 에너미즈(Enemies) 등의 프로젝트를 제작한 팀이 2024년에 개발한 실시간 시네마틱 데모입니다.

이 프로젝트는 APV(적응적 프로브 볼륨)와 시나리오 블렌딩으로 개선된 조명부터 GPU 상주 드로어를 통한 성능 개선까지, HDRP에서 최근에 어떤 발전이 있었는지 보여 줍니다. 시간의 유령은 ECS(엔티티 컴포넌트 시스템) 아키텍처 기반의 Unity Entities 패키지를 포함하는 Unity의 DOTS(데이터 지향 기술 스택)를 활용하여 최대 1,200만 개의 초목 인스턴스를 렌더링합니다.



시간의 유령에서는 HDRP를 사용하여 사실적인 조명을 만들었습니다.



Unity 6의 새로운 **GPU 상주 드로어**는 최소한의 CPU 오버헤드로 데모의 방대한 환경을 렌더링할 수 있도록 배칭 및 인스턴싱 효율을 개선합니다.

시간의 유령에는 **VFX Graph**와 **Shader Graph**도 활용되었습니다. 개발팀은 이 두 가지 툴을 사용하여 사실적인 폭발, 다이내믹 안개, 정교한 환경 효과를 만들었습니다.



시간의 유령의 비주얼 효과에는 VFX Graph가 사용되었습니다.

시간의 유령은 [여기](#)에서 시청하실 수 있습니다. 유나이티드 바르셀로나 2024의 [제작 비하인드 세션](#)에서는 조명, 환경, 캐릭터 개발에 대한 Unity 오리지널 팀의 접근 방식을 자세히 알아볼 수 있습니다.

에셋 스토어에 있는 2개의 샘플 패키지를 통해 시간의 유령에 사용된 아트와 기술을 살펴볼 수 있습니다.

- **Time Ghost: Environment:** 이 패키지는 **DOTS ECS**와 절차적 툴을 사용하여 대량의 씬 엔티티를 관리하고 리소스를 할당합니다.
- **Time Ghost: Character:** 데모의 주인공에서 **Sentis AI**를 사용한 실시간 캐릭터 애니메이션과 물리 기반 인터랙션을 확인할 수 있습니다.



에셋을 다운로드하여 Unity 6 프로젝트로 임포트하면 시간의 유령에 사용된 고급 기법을 직접 살펴볼 수 있습니다.



시간의 유령 메인 캐릭터는 다운로드할 수 있습니다.

Unity 6.1의 HDRP 업데이트

Unity 6.1의 HDRP에는 여러 새로운 기능과 개선 사항이 도입되었습니다. 2022 LTS의 HDRP(본 전자책의 이전 버전에 사용된 버전)에서 전환할 계획이라면 아래에 정리된 내용을 살펴보세요.

Lighting

LightBaker: HDRP의 새로운 백엔드 LightBaker v1.0은 모든 프레임에서 씬 변경이 있는지 지속적으로 확인하는 대신 베이킹을 시작할 때 씬 상태의 ‘스냅샷’을 찍습니다. 베이킹은 수동으로 취소하지 않는 한 중단 없이 실행되므로 불필요한 재계산이 줄어들고 에디터 성능이 향상됩니다.

베이킹 프로파일: 이제 Lighting 창에 GPU 라이트매퍼에 대한 [GPU 베이킹 프로파일](#)이 제공됩니다. 이 기능은 라이트맵을 타일로 구분하며, 빠른 베이킹과 낮은 메모리 사용량 중 하나에 최적화되는 프리셋을 사용할 수 있습니다.

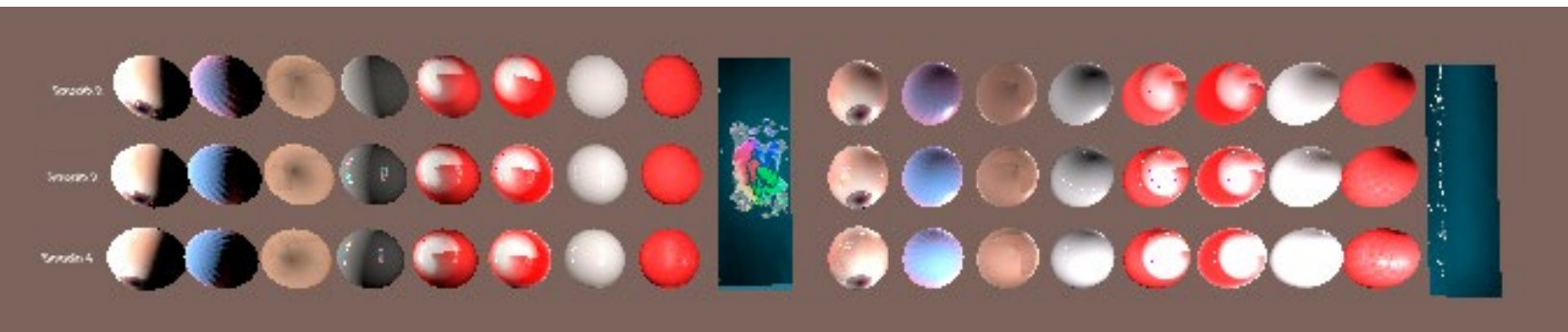
APV(적응적 프로브 볼륨) 업데이트: 이제 LightBaker에서 라이트맵과 독립적으로 라이트 프로브와 APV를 베이킹할 수 있습니다. APV는 [스카이 오클루전](#) 베이킹도 지원하므로, 게임 오브젝트가 [앰비언트 프로브](#)의 하늘 색상을 사용하여 동적으로 조명을 조정할 수 있습니다. 따라서 APV를 사용하여 낮-밤 사이클을 시뮬레이션할 수 있습니다.

레이트레이싱 개선: 레이트레이싱에 레이트레이싱 가속 구조를 더 세밀하게 제어할 수 있도록 새로운 API가 추가되었습니다. [RayTracingAccelerationStructure](#)에 다음과 같은 새로운 메서드가 추가되었습니다.

- [UpdateInstanceGeometry](#): 새로운 [RayTracingMode.DynamicGeometryManualUpdate](#) 모드를 사용하여 **BLAS(Bottom-Level Acceleration Structure)**를 수동으로 업데이트할 수 있습니다. 이렇게 하면 모든 프레임에서 지오메트리가 자동으로 업데이트되는 대신, 특정 인스턴스를 ‘더티’로 설정한 다음 가속 구조를 다시 구축하는 작업을 수동으로 트리거할 수 있습니다. 이를 통해 레이트레이싱 씬에서 동적 오브젝트의 성능을 개선할 수 있습니다.
- [AddInstancesIndirect](#): 변환 매트릭스에 GraphicsBuffer를 사용하여 **RTAS(Ray Tracing Acceleration Structure)**에 여러 개의 레이트레이싱 인스턴스를 추가합니다. 이렇게 하면 해당 인스턴스를 GPU가 관리하게 되어 CPU 오버헤드가 줄어들고 성능이 개선됩니다. 인스턴스는 메시 기반일 수도 있고 절차적일 수도 있습니다.
- [RemoveInstances](#): LayerMasks 또는 레이트레이싱 모드를 기반으로 레이트레이싱 인스턴스를 제거합니다. 런타임에 씬이 변경됨에 따라 가속 구조를 동적으로 업데이트하여 성능을 최적화할 수 있습니다.
- [CullInstances](#): 필터링 및 컬링 파라미터를 추가합니다. 특정 기준에 따라 쉽게 가속 구조를 업데이트할 수 있습니다.

라이트맵을 위한 바이큐빅 샘플링: 이 기능은 라이트맵을 불러 처리하고 매끄럽게 만들어서 저해상도에서도 고품질로 보이도록 합니다. 베이킹된 광원을 사용하며 메모리와 성능 제약으로 인해 저해상도 라이트맵이 요구될 수 있는 대규모 환경에서 유용합니다. **Project Settings > Graphics > Use Bicubic Lightmap Sampling**에서 활성화하세요.

면 광원 개선: 이제 면 광원이 범위 감쇠에 ‘pillow’ 원도잉 함수를 사용합니다. 따라서 면 광원이 더 부드럽게 페이드 아웃되고 다양한 유형의 머티리얼에서 더 원활하게 작동하며, CPU 메모리 사용량은 줄어듭니다.



면 광원이 범위 감쇠에 ‘pillow’ 원도잉 함수를 사용합니다.

패스트레이싱 개선: 디스크 및 튜브 면 광원에서 이제 [패스트레이싱](#)이 지원됩니다. **상자 광원**도 올바른 조명 동작을 보이도록 수정되었습니다. 이에 더해 면 광원에 **볼류메트릭 안개 컨트롤**이 추가되었습니다. 이를 통해 더 사실적인 대기 효과를 보이도록 안개를 미세 조정할 수 있습니다.

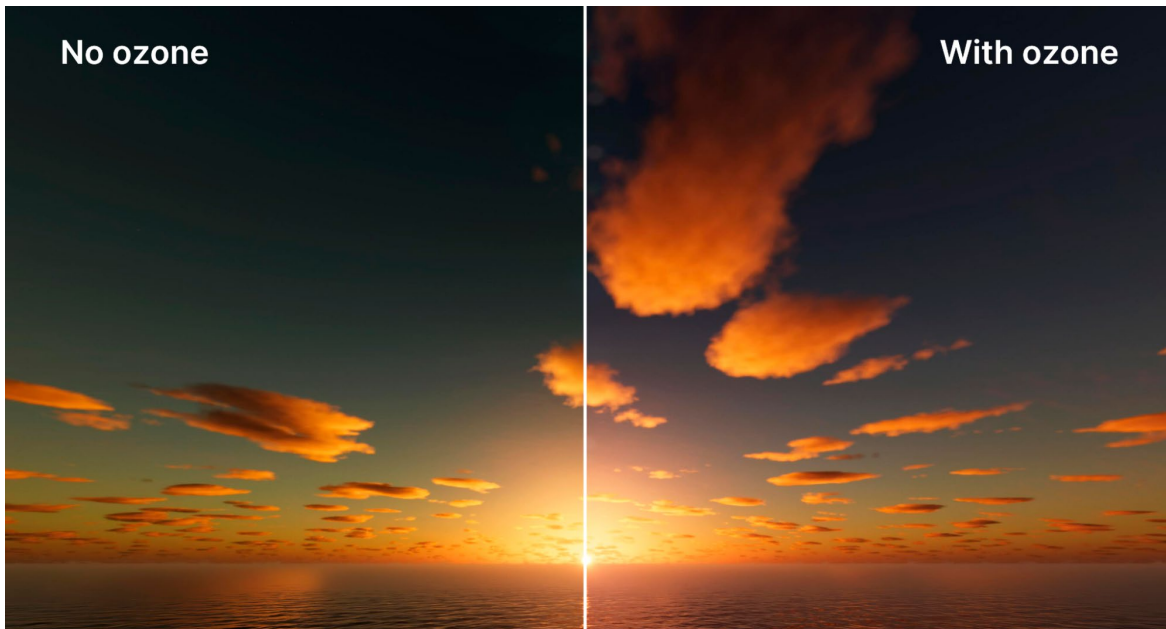


디스크 및 튜브 면 광원에서 패스트레이싱이 지원됩니다.

환경 효과

행성 파라미터: 이제 [Visual Environment 오버라이드](#)의 공유 파라미터 세트가 볼류메트릭 구름, 안개, 물리 기반 하늘과 같은 다양한 환경 효과에 영향을 줍니다.

Physically Based Sky Volume: 사전 연산 단계가 제거되어 [PBR Sky Volume](#)이 최적화되었습니다. 이에 따라 성능에 최소한의 영향을 주면서 하늘이 실시간으로 업데이트됩니다. 오존 레이어가 새로 추가되어 수평선 부근에서 하늘 색상 정확도가 향상되면서 한층 사실적인 황혼 효과가 구현됩니다.



오존 레이어를 통해 수평선 부근의 색상 정확도를 향상할 수 있습니다.

항공 시점에서는 산이나 구름처럼 멀리 있는 오브젝트를 볼 때 대기 중 입자에 의한 빛 흡수를 시뮬레이션할 수 있습니다.



Physically Based Sky Volume을 통해 대기 산란을 시뮬레이션할 수 있습니다.

볼류메트릭 구름: 구름이 더 이상 원거리 평면에 의해 잘리지 않으며, 지구 곡률 컨트롤이 Visual Environment 설정으로 이동되었습니다.

물 시스템: 물 시스템에서 수면 위의 게임 오브젝트 트랜스폼(이동, 회전, 음수 스케일링) 지원이 향상되었습니다. Unity 6.1에서는 Shader Graph가 거품과 수평 수면 변형(예: 밀려오는 파도)을 출력할 수 있도록 Water Decal 타겟이 도입되었습니다.

볼류메트릭 안개: 볼류메트릭 조명 성능이 최적화되었습니다.

성능 및 최적화

다이내믹 해상도: Unity 6.1에는 **STP(시공간 포스트 프로세싱)**가 도입되었습니다. 이는 (주변 픽셀을 사용하여 디테일을 개선하는) 공간 업스케일링과 (이전 프레임을 사용하여 예지를 개선하는) 시간 업스케일링이 결합된 새로운 고품질 이미지 업스케일링 기법입니다.

GPU 상주 드로어: 새로운 **GPU 상주 드로어**는 배치 렌더 그룹 API를 사용하여 복잡한 씬을 렌더링할 때 성능을 크게 개선합니다.

VRS(Variable Rate Shading): 이제 VRS(Variable Rate Shading)가 커스텀 패스를 지원합니다. 특정 화면 영역에서 셰이딩 해상도가 동적으로 조정되어 비주얼 품질은 그대로 유지되면서 GPU 워크로드가 줄어듭니다.

패키지 관리자 샘플

패키지 관리자의 [HDRP 샘플 프로젝트](#)가 재작업을 거쳐 일관성을 한층 더 높였습니다. 설명이 개선되었고, 리소스가 여러 파이프라인에서 공유됩니다.

- **Lit Material 샘플:** 이 씬의 모든 머티리얼에서는 **HDRP/릿 셰이더**를 사용하여 각종 **머티리얼 유형** (표준, 피하 산란, 이방성 등)을 확인할 수 있습니다. 각 샘플은 HDRP가 다양한 표면에서 사실적인 조명을 처리하는 방식을 나타냅니다.
- **Transparent Material 샘플:** 이 씬에서는 굴절 머티리얼, 투명 오브젝트 스택킹, 투명 머티리얼의 그림자 등을 통해 HDRP가 투명도를 처리하는 방식을 살펴볼 수 있습니다. 래스터화, 레이트레이싱, 패스트레이싱을 전환하며 다양한 렌더링 기법을 살펴볼 수 있습니다.
- **Volumetric 샘플:** 이 씬에서는 볼류메트릭 안개, 3D 텍스처, Shader Graph 효과 및 블렌딩 모드를 확인할 수 있습니다. 샘플에서 여러 안개 효과, 절차적 노이즈 및 연기 요소를 재현하는 방법을 볼 수 있습니다.
- **Environment 샘플:** 이 씬에서는 섬 터레인과 바다 물 시스템 위의 각종 구름 설정을 살펴볼 수 있습니다.
- **Water 샘플:** 이 프로젝트에는 새로운 물 **Cave** 샘플 씬이 있습니다. 씬에서는 물 시스템 커스텀 버퍼를 샘플링하는 커스텀 패스와 로컬 볼류메트릭 안개가 사용되어 고급 물 렌더링 기법을 보여 줍니다. **Rolling Wave** 씬은 수평 변형을 구현하여 밀려오는 파도와 같은 효과를 만드는 방법을 보여 줍니다.

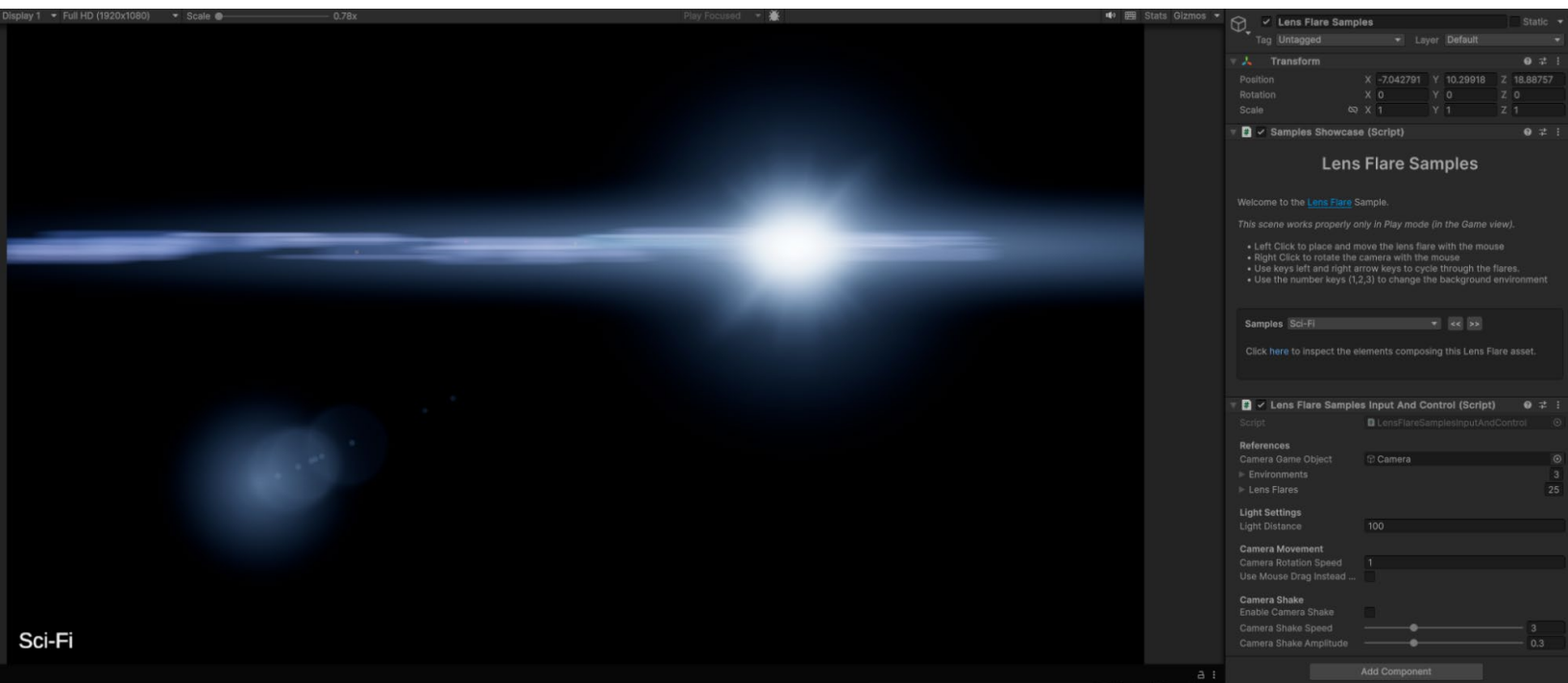


패키지 관리자에는 다양한 샘플 패키지가 들어 있습니다.

렌즈 플레어

Unity 6에서는 비주얼 충실도와 워크플로 개선을 위해 **렌즈 플레어**가 여러 새로운 기능으로 개선되었습니다.

- **향상된 XR 지원:** 렌즈 플레어의 XR 애플리케이션 호환성이 개선되어 이제 다양한 기기에서 일관적인 플레어 효과를 구현할 수 있습니다.
- **Ring Procedural Shape:** 새로운 링 모양 절차적 플레어를 사용하여 밝은 광원에 유용한 동심원 구조의 플레어를 만들 수 있습니다.
- **재귀적 렌즈 플레어 에셋:** 이제 렌즈 플레어 에셋이 다른 렌즈 플레어의 요소가 될 수 있습니다. 이에 따라 한층 복잡하고 레이어링된 플레어 효과를 만들 수 있습니다.
- **방사형 그레디언트 지원:** 이제 절차적 셰이프가 방사형 그레디언트를 지원하므로, 한층 더 자연스러워 보이는 플레어를 만들 수 있습니다.
- **Light Override 기능:** 새로운 Light Override 기능을 사용하여 하나의 광원으로 여러 개의 플레어가 영향을 받도록 할 수 있습니다. 이에 따라 여러 광원의 렌즈 플레어 효과 설정이 간소화됩니다.



Unity 6에서 렌즈 플레어가 개선되었습니다.

모발

이제 **High-Quality Line Rendering 볼륨 오버라이드**로 모발 너비를 버텍스당 센티미터 단위로 정의할 수 있습니다. 이에 따라 모발 제어와 정확도가 향상됩니다.



버텍스당 너비 값으로 제어와 정확도를 향상시킬 수 있습니다.

다음은 균일한 0.01cm 너비로 얼굴의 솜털을 구현한 예입니다.



버텍스당 너비 값을 균일한 0.01cm 너비로 설정하여 솜털을 구현할 수 있습니다.

이에 더해 새로운 스크린 커버리지 기반 LOD 모드가 뷰포트 가시성을 기반으로 모발 가닥을 동적으로 컬링합니다. 아티스트는 애니메이션 커브를 사용하여 가닥 밀도를 제어할 수 있습니다. X축은 스크린 공간 점유율을 나타내고, Y축은 렌더링되는 가닥의 비율을 정의합니다. 이로 인해 성능과 비주얼 품질이 향상됩니다.

HDRP 설치

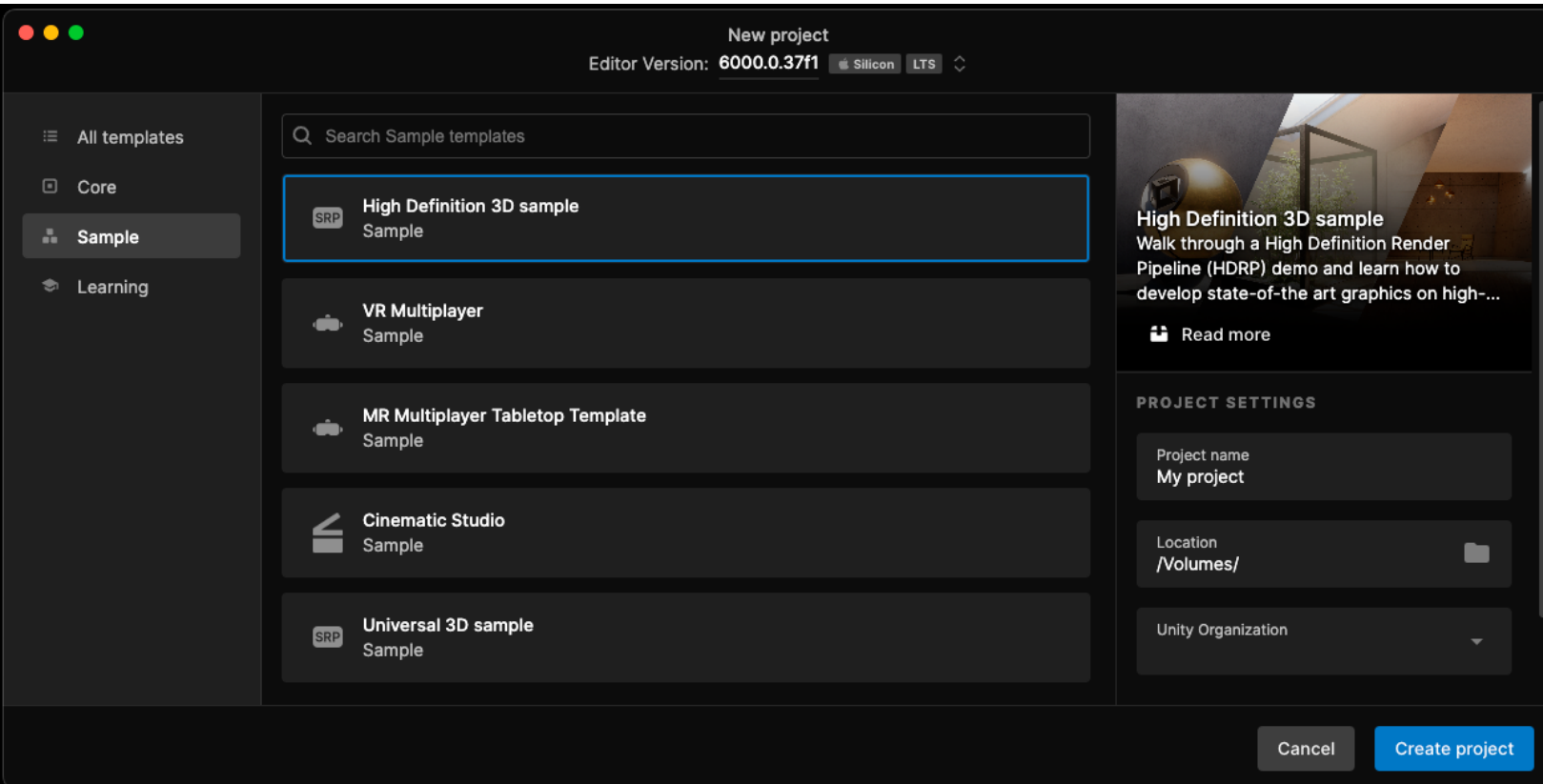
프로젝트에서 HDRP를 설정하려면 **HDRP 템플릿 프로젝트**로 시작하거나 **High Definition RP 패키지**를 수동으로 설치합니다.

HDRP 템플릿은 물리 기반 조명 설정, 데칼, 볼륨, 높은 정확도의 머티리얼이 들어 있어 하이엔드 비주얼 프로젝트를 위한 좋은 출발점이 됩니다.

템플릿을 사용하여 HDRP 프로젝트 만들기

Unity Hub를 열고 **프로젝트** 탭으로 이동합니다. 새 **프로젝트**를 클릭하고 **샘플** 탭으로 이동한 다음 템플릿으로 **High Definition 3D Sample Scene**을 선택합니다.

프로젝트 이름을 입력하고 **프로젝트 생성**을 클릭합니다. HDRP와 종속성이 자동으로 설치됩니다.



Hub에서 HDRP 프로젝트를 만듭니다.

수동으로 HDRP 설치

기존 프로젝트에 HDRP를 추가하려면 Unity 프로젝트를 열고 **Window > Package Manager**로 이동합니다.

Package Manager의 드롭다운 메뉴에서 **Unity Registry**를 선택합니다. 패키지 목록에서 **High Definition RP**를 찾아서 선택합니다.

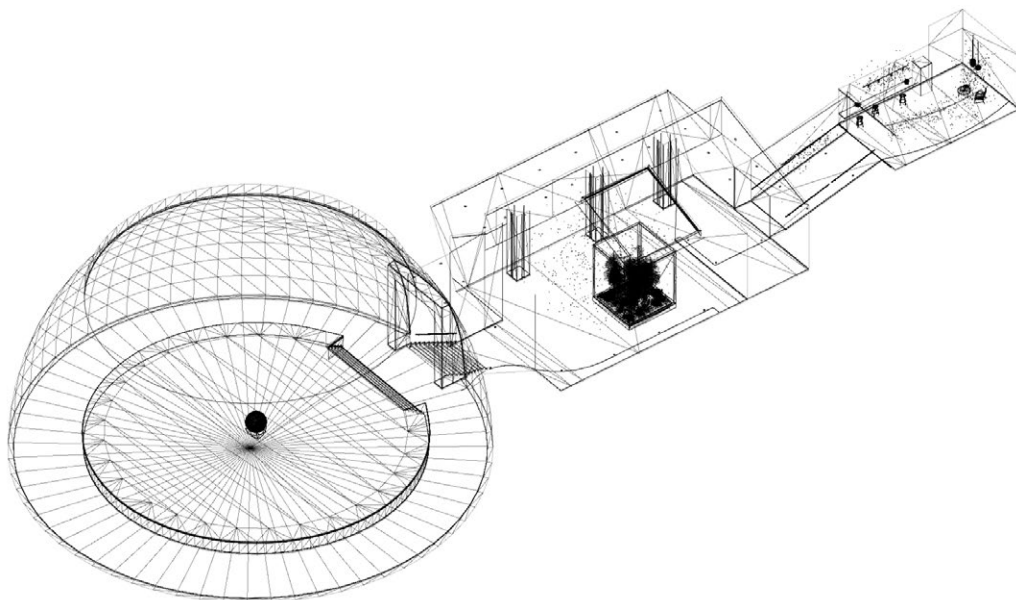
Install을 클릭합니다. 오류가 발생할 경우, HDRP 마법사를 사용하여 설정 문제를 수정합니다.

프로젝트에 HDRP를 설정하는 방법에 대한 자세한 내용은 기술 자료의 [고해상도 렌더 파이프라인 시작하기](#)를 참조하세요.

HDRP 샘플 씬

Unity Hub에서 제공하는 3D 샘플 씬은 HDRP를 시작하는 데 도움이 되는 템플릿 프로젝트입니다. 이 가벼운 프로젝트에는 언제든지 빠르게 로드하여 참조할 수 있는 작은 게임 레벨이 들어 있습니다.

이 가이드에서는 주기적으로 이 프로젝트를 사용하여 HDRP의 많은 기능을 살펴볼 것입니다.



3D 샘플 씬 환경을 보여 주는 와이어프레임



작은 공간이 여러 개 포함된 환경으로, 세 개의 공간에 조명이 각자 다르게 설정되어 있습니다. 방향 광원은 태양을 나타내며, 강도는 실제와 똑같은 10만 렉스입니다. 각 공간마다 조명 환경에 맞게 카메라의 노출이 조정됩니다. WASD 키와 마우스를 사용하여 레벨에서 FPS Controller를 사용할 수 있습니다.

- **첫 번째 공간**은 위에서 햇빛이 비치는 둥근 플랫폼입니다. 데칼을 통해 콘크리트 바닥에 먼지와 물웅덩이를 추가했습니다.
- **두 번째 공간**에서는 채광창에서 쏟아지는 입체적인 빛줄기와 유리 케이스 안에 있는 초목 등의 고급 머티리얼을 확인할 수 있습니다.
- **세 번째 공간**에는 실내 인공조명과 발광 머티리얼 등이 있습니다.



샘플 씬은 세 개의 공간으로 구성되어 있습니다.

템플릿 씬에 대해 상세하게 설명하는 [이 블로그 게시물](#)에서 HDRP 3D 샘플 씬에 대해 자세히 알아보세요.

더 많은 HDRP 샘플 콘텐츠

샘플을 모두 살펴보았다면 이제 다른 유용한 프로젝트를 살펴볼 차례입니다.

Time Ghost: Environment 샘플 패키지에는 Unity 시네마틱 데모 **시간의 유령**을 위해 생성된 환경이 들어 있습니다. Unity 오리지널 팀이 DOTs의 Entities 패키지와 그 밖의 툴을 사용하여 이 환경 씬을 만든 방식을 자세히 살펴볼 수 있습니다. 데모에서는 APV 조명, 시나리오 블렌딩, GPU 상주 드로어 등 Unity 6와 HDRP의 최신 개선 사항이 사용되었습니다.



시간의 유령 환경

Time Ghost: Character 샘플 패키지에는 **시간의 유령**의 캐릭터가 등장합니다. 이 캐릭터가 천 변형을 위한 머신러닝 모델을 보여 주며 **더 헤라티크**와 **에너미즈**에서 볼 수 있는 모발 시스템을 활용합니다. 이 **Unity Discussions** 게시물에서 **시간의 유령 제작 비하인드**에 대해 알아보세요.



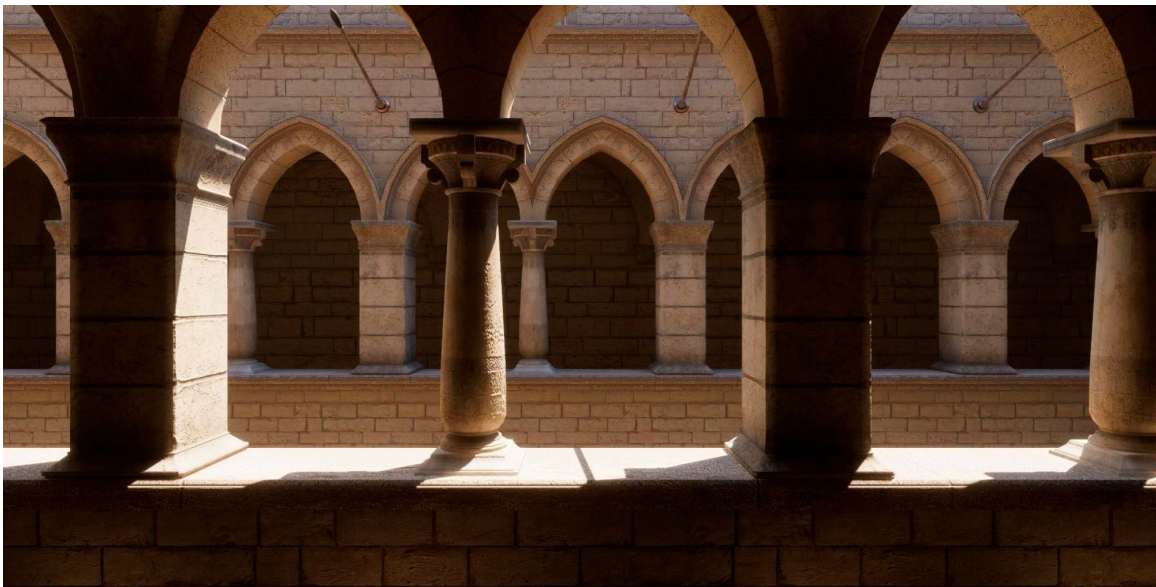
시간의 유령 캐릭터의 천 구현입니다.

에너지 데모 프로젝트는 디지털 휴먼 캐릭터뿐만 아니라 실제로 착각할 만큼 사실적인 배경 환경 애니메이션을 선보입니다. 이 데모는 프로브 볼륨과 레이트레이싱 효과를 사용하고, NVIDIA의 DLSS(딥러닝 슈퍼샘플링) 네이티브 지원을 통합하여 4K 해상도로 실행됩니다.



에너지 데모 프로젝트에는 정성을 들여 제작한 멋진 인테리어가 담겨 있습니다.

스폰자 궁전 아트리움(Sponza Palace Atrium)은 그래픽스 프로그래머와 아티스트들이 널리 사용하는 데모입니다. 실내와 실외 공간이 모두 있어 조명을 테스트하기에 이상적인 환경입니다. 이 버전은 HDRP로 리마스터되었습니다.



스폰자 아트리움



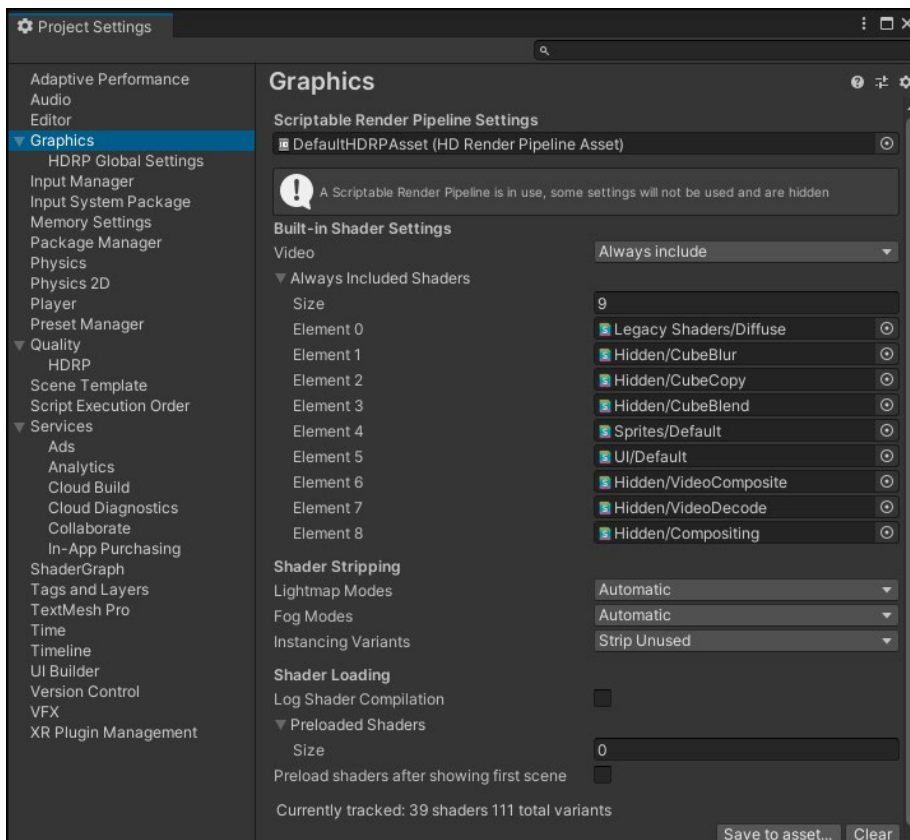
에셋 스토어에서 [Cinematic Studio Template](#)을 다운로드해 시네마틱 영상이나 애니메이션 영화를 만드는 방법을 알아보세요. 이 템플릿에서는 스타일라이즈드 렌더링과 사실적 렌더링을 함께 사용한 Mich-L이라는 재미있는 단편 영화를 통해 장면과 조명을 어떻게 설정했는지 알아볼 수 있습니다.



시네마틱 스튜디오 샘플

프로젝트 설정

Project Settings(Edit > Project Settings)에 있는 **Graphics, HDRP Global Settings, Quality**에는 여러 필수 설정이 있습니다.



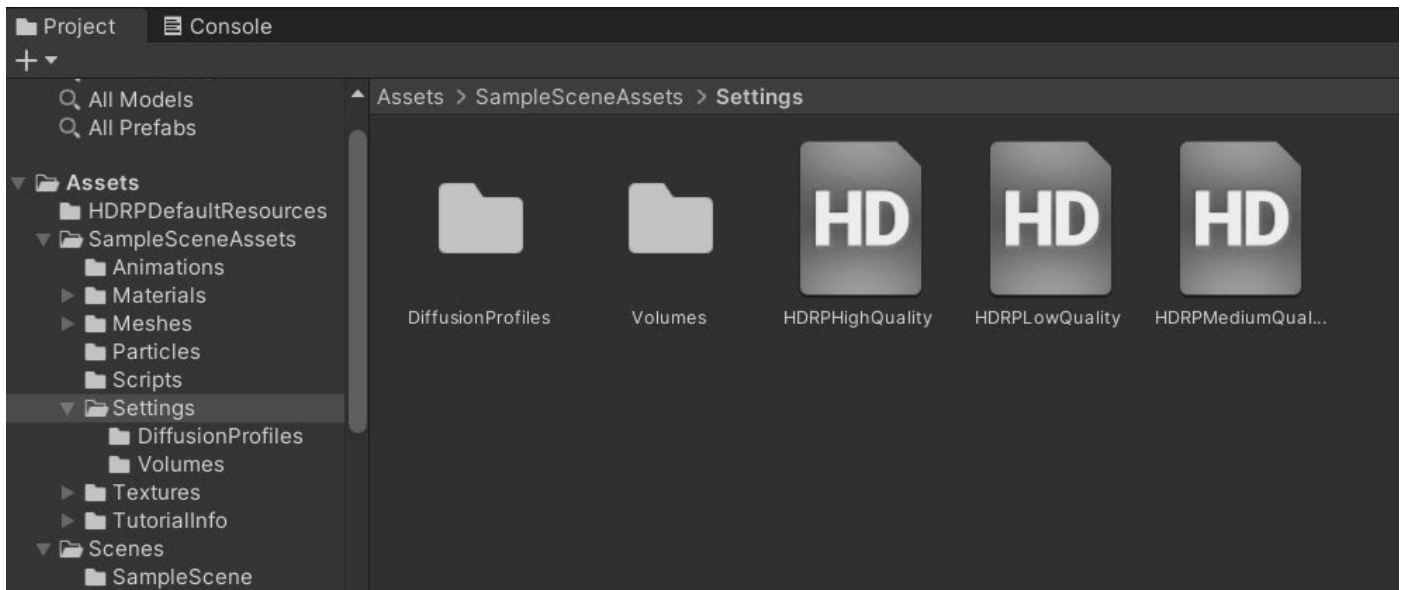
프로젝트 설정

그래픽스 설정

상단의 **Scriptable Render Pipeline Settings** 필드에는 모든 HDRP 설정을 보관하는 파일이 표시됩니다.

프로젝트별로 여러 **파이프라인 에셋**이 있을 수 있는데, 각 파이프라인 에셋을 별도의 설정 파일이라고 생각하면 됩니다. 예를 들면, 파이프라인 에셋을 사용하여 여러 타겟 플랫폼에 대한 전용 설정을 저장하거나 플레이어가 런타임에 바꿀 수 있는 여러 화질 수준을 나타낼 수도 있습니다.

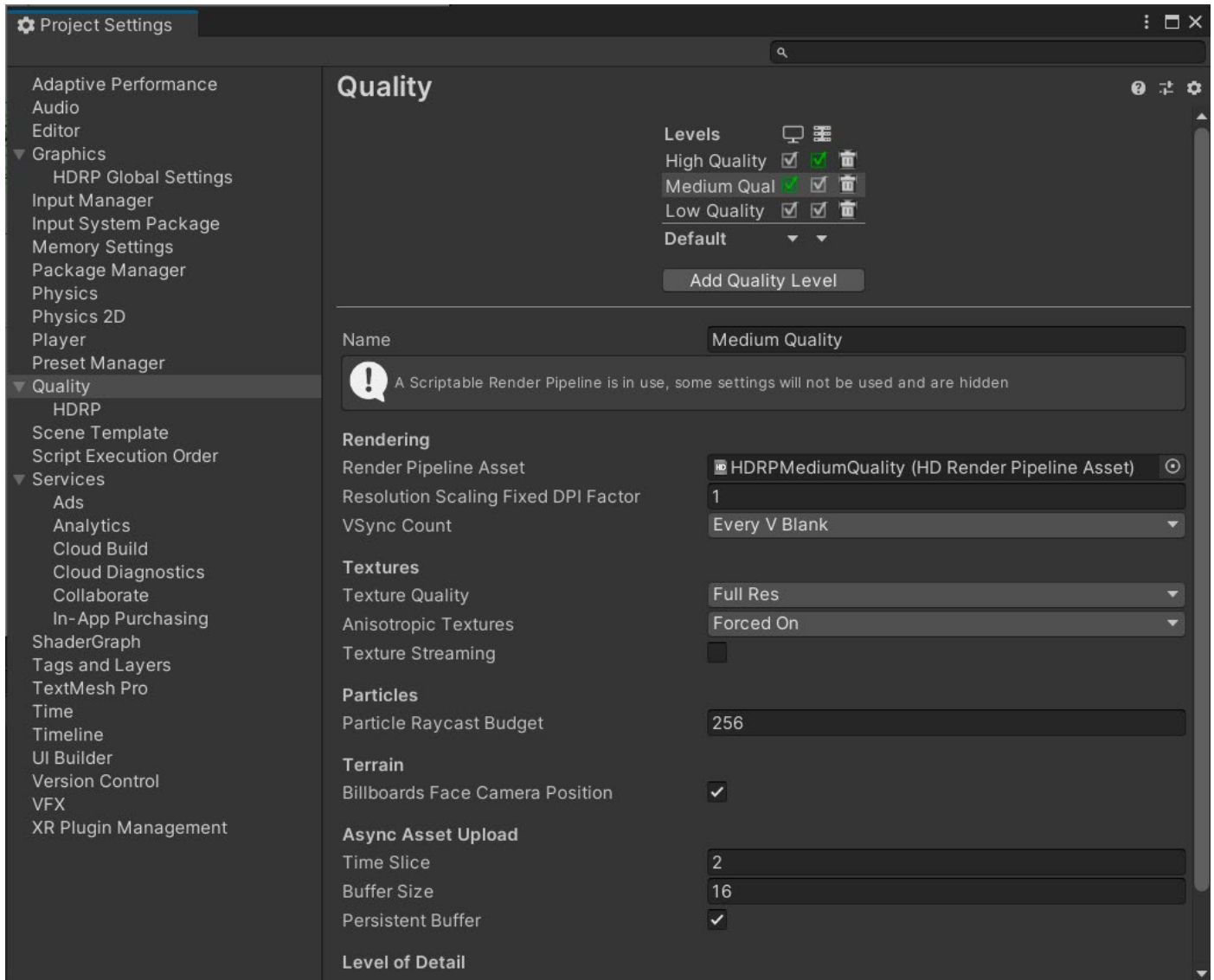
3D 샘플 씬의 경우 Settings 폴더에 **HDRPHighQuality**, **HDRPLowQuality**, **HDRPMediumQuality** 등의 파이프라인 에셋이 포함되어 있습니다. 또한 **DefaultHDRPAsset**이 포함된 **HDRPDefaultResources** 폴더도 있습니다.



3D 샘플 씬에는 저품질, 보통 품질, 고품질 파이프라인 에셋이 포함되어 있습니다.

품질 설정

품질 설정을 사용하면 파이프라인 에셋 중 하나를 사전 정의된 품질 수준에 맞출 수 있습니다. 상단의 **Level** 중 하나를 선택하면 **Rendering** 옵션에 표시된 특정 **렌더 파이프라인 에셋**이 활성화됩니다.

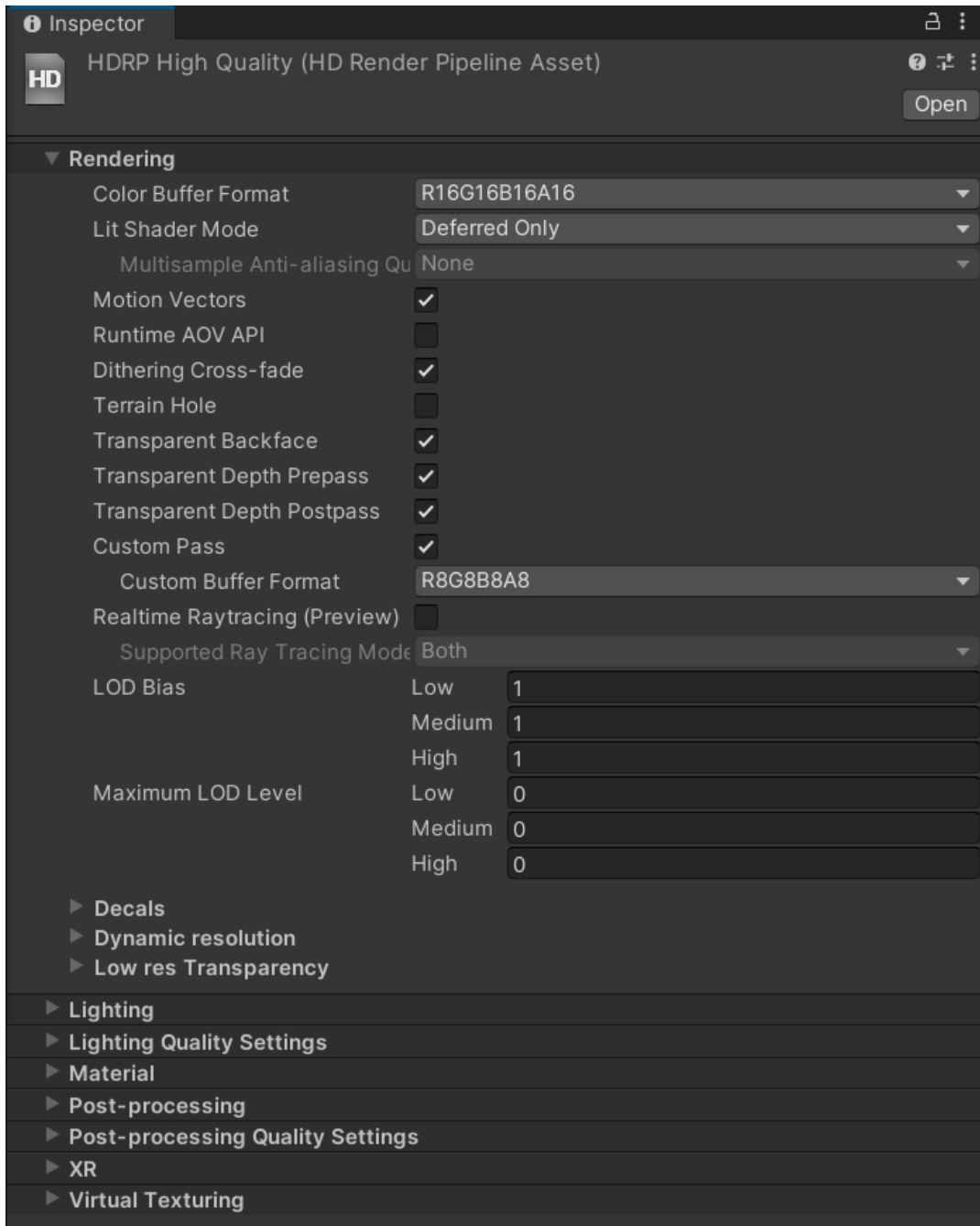


파이프라인 에셋을 활성화하는 상단의 품질 수준입니다.

기본값을 커스터마이징하거나 파이프라인 에셋과 품질 수준을 추가할 수도 있습니다.

품질 수준은 파이프라인에서 작동하는 특정 시각적 기능을 나타냅니다. 예를 들어 애플리케이션에 여러 그래픽스 티어를 생성해 두면, 플레이어가 런타임 중에 하드웨어에 맞춰 사용할 품질 수준을 선택할 수 있습니다.

실제 파이프라인 설정은 **Quality/HDRP** 하위 섹션에서 편집합니다. 프로젝트(Project) 뷰에서 파이프라인 에셋을 선택하고 인스펙터(Inspector)에서 설정을 편집할 수도 있습니다.



파이프라인 에셋 편집



HDRP 최적화

파이프라인 에셋에서 활성화하는 기능이 많을수록 리소스를 더 많이 소모하게 됩니다. 일반적으로 의도한 효과를 구현하는 데 필요한 기능만 사용하도록 프로젝트를 최적화해야 합니다. 반드시 필요하지 않은 기능은 비활성화해 두면 성능을 개선하고 리소스를 아낄 수 있습니다.

사용하지 않을 경우 비활성화할 만한 대표적인 기능을 몇 가지 소개합니다.

- **HDRP 에셋**에서 Decals, Low-res Transparency, Transparent Backface/Depth Prepass/Depth Postpass, SSAO, SSR, Contact Shadows, Volumetrics, Subsurface Scattering, Distortions를 모두 비활성화할 수 있습니다.
- 카메라의 **Frame Settings**(메인 카메라, 반사 등의 통합된 효과에 사용되는 카메라, 커스텀 효과에 사용되는 추가 카메라)에서 Refraction, Post-Process, After Post-Process, Transmission, Reflection Probe, Planar Reflection Probe, Big Tile Prepass를 모두 비활성화할 수 있습니다.

이 [블로그 게시물](#)에서 성능을 향상하기 위한 HDRP 설정을 알아보세요.

HDRP 전역 설정

HDRP Global Settings 섹션(또는 버전 12 이전의 경우 **HDRP Default Settings**)은 프로젝트의 기준 설정을 정합니다. 이러한 설정은 씬에서 카메라의 위치에 따라 로컬이나 글로벌 볼륨 컴포넌트를 배치하여 오버라이드할 수 있습니다(아래 볼륨 참조).

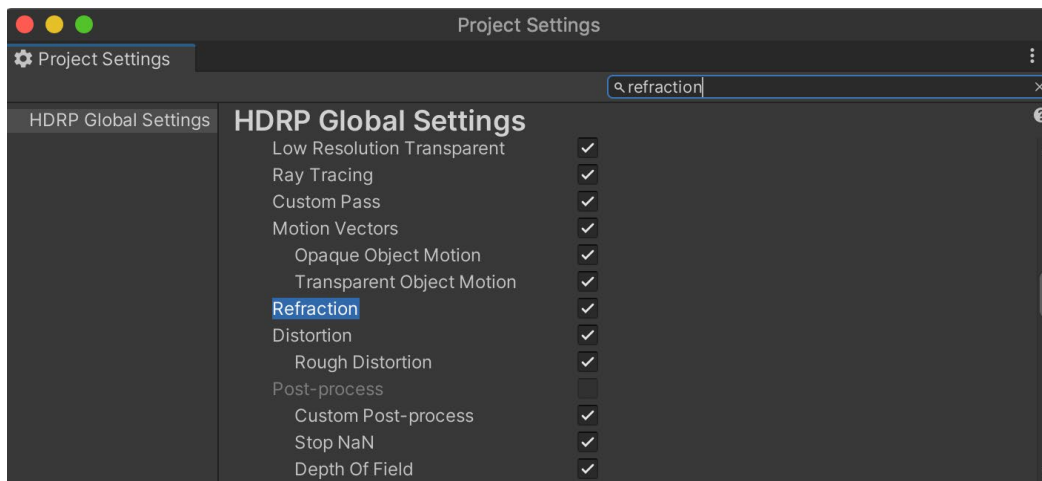
Global Settings는 상단 필드에 정의된 별도의 파이프라인 에셋에 저장됩니다. 여기에서 기본 렌더링과 포스트 프로세싱 옵션을 설정할 수 있습니다.

HDRP 기능 활성화

프로젝트 개발 중에 특정 기능을 토글하기 위해 **Global Settings**를 다시 확인해야 하는 경우도 있습니다. 일부 기능의 경우 **HDRP Global Settings**에 있는 체크박스가 활성화되어 있지 않으면 렌더링되지 않습니다.

렌더링 성능과 메모리 사용량에 부정적인 영향을 줄 수 있으므로 필요한 기능만 활성화해야 합니다. 또한 용도에 따라 **Volume Profiles**에 표시되는 설정이 있고, 어떤 기능은 **Frame Settings**에 표시됩니다.

HDRP 기능에 아직 익숙하지 않다면 Project Settings 오른쪽 상단에 있는 검색 필드를 사용하세요. 검색 필드를 사용하면 관련 패널만 검색어를 강조 표시하여 볼 수 있습니다.



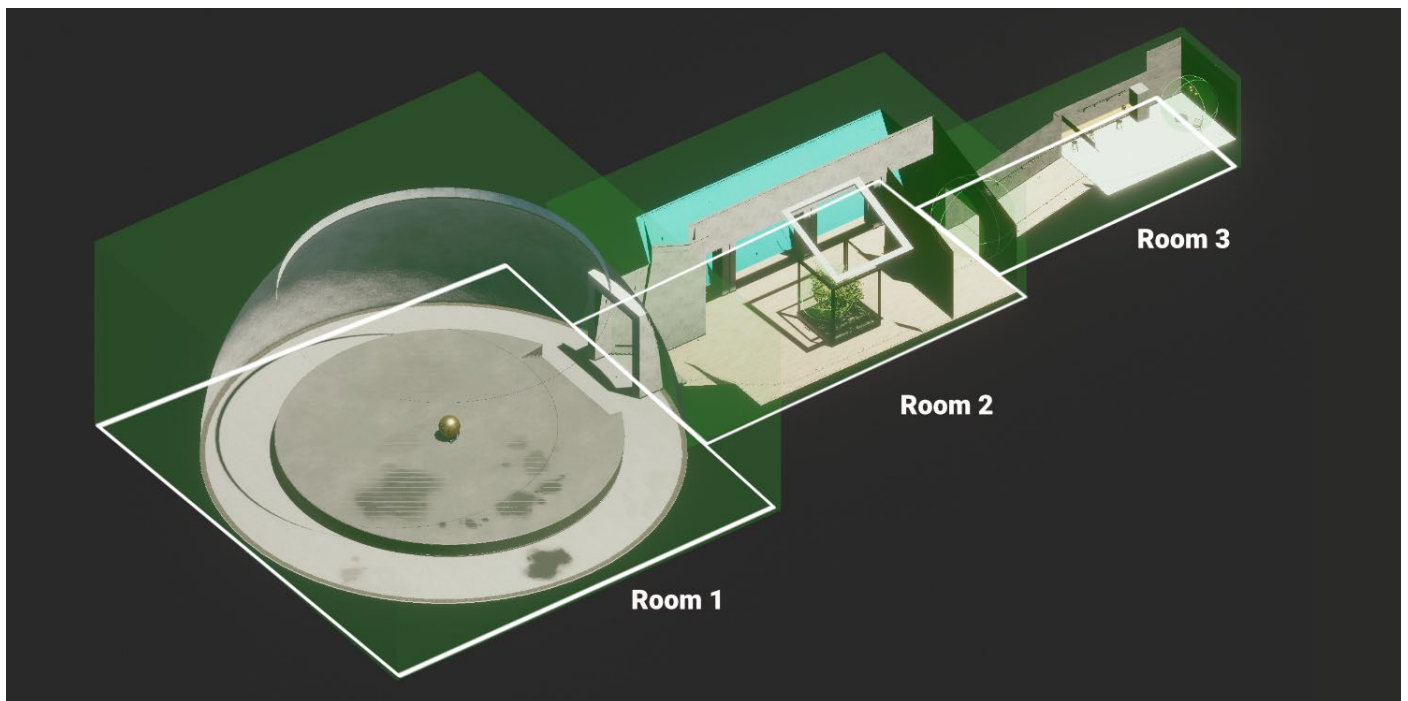
HDRP 기능을 검색합니다.

HDRP Global Settings에서 활성화한 기능이 모든 카메라에서 항상 렌더링되는 것은 아닙니다. **Projects Settings > Quality**에서 품질 수준을 선택한 렌더 파이프라인 에셋에서도 해당 기능을 지원하도록 설정해야 합니다.

예를 들어 카메라가 볼류메트릭 구름을 렌더링하도록 하려면 **HDRP Global Settings > Frame Settings > Camera > Lighting**에서, 그리고 활용 중인 렌더 파이프라인 에셋의 **Lighting > Volumetrics**에서도 Volumetric Clouds 옵션을 토글해야 합니다.

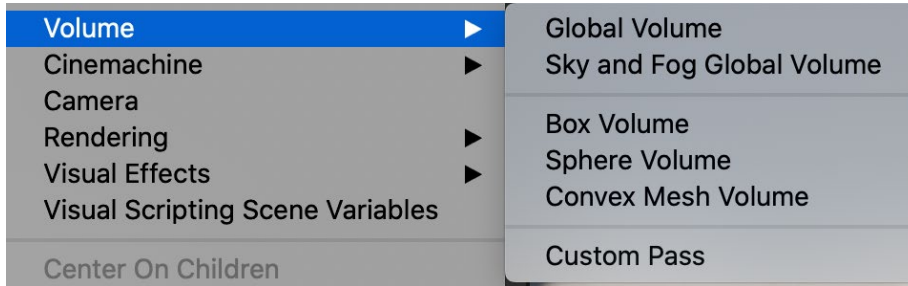
볼륨

HDRP는 **볼륨 프레임워크**를 사용합니다. 이 시스템을 활용해 씬을 분할하고 카메라 위치에 따라 특정 설정이나 기능을 사용할 수 있습니다. 예를 들어, HDRP 템플릿 레벨에는 각기 다른 조명 설정이 되어 있는 3개의 구역이 있습니다. 따라서 서로 다른 볼륨이 각 공간을 둘러싸고 있습니다.



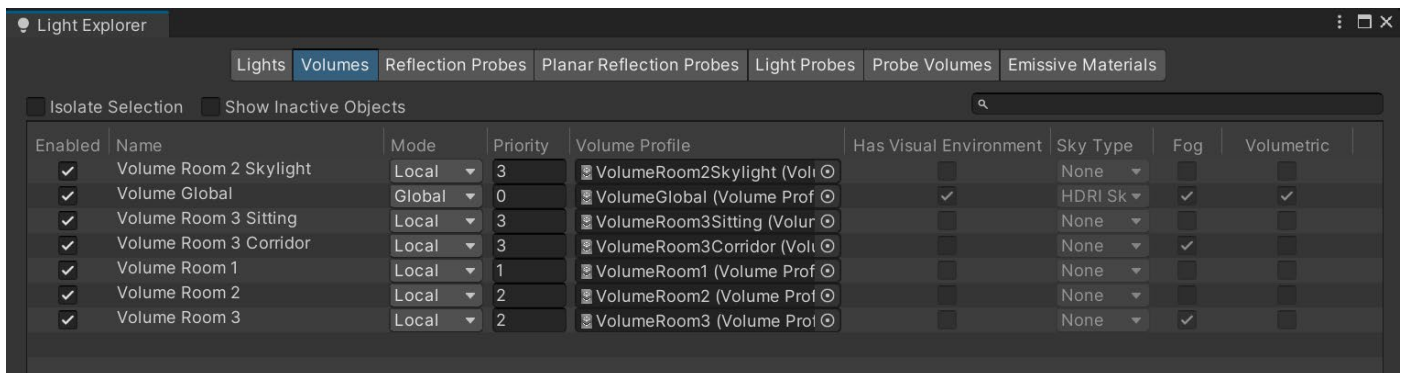
조명 조건이 서로 다른 공간이 볼륨으로 덮여 있습니다.

볼륨은 볼륨 컴포넌트가 있는 플레이스홀더 오브젝트입니다. **GameObject > Volume** 메뉴에서 프리셋을 선택하여 볼륨을 만들 수 있습니다. 아니면 게임 오브젝트를 만들고 수동으로 해당 컴포넌트를 추가하면 됩니다.



프리셋을 사용한 볼륨 오브젝트 생성

볼륨 컴포넌트는 어느 게임 오브젝트에나 추가할 수 있기 때문에 계층(Hierarchy) 창에서 찾기 어려울 수 있습니다. Light Explorer(**Window > Rendering > Light Explorer > Volumes**)를 사용하면 로드된 씬에서 볼륨을 쉽게 찾을 수 있습니다. 이 인터페이스를 사용하여 빠르게 볼륨을 조정할 수 있습니다.

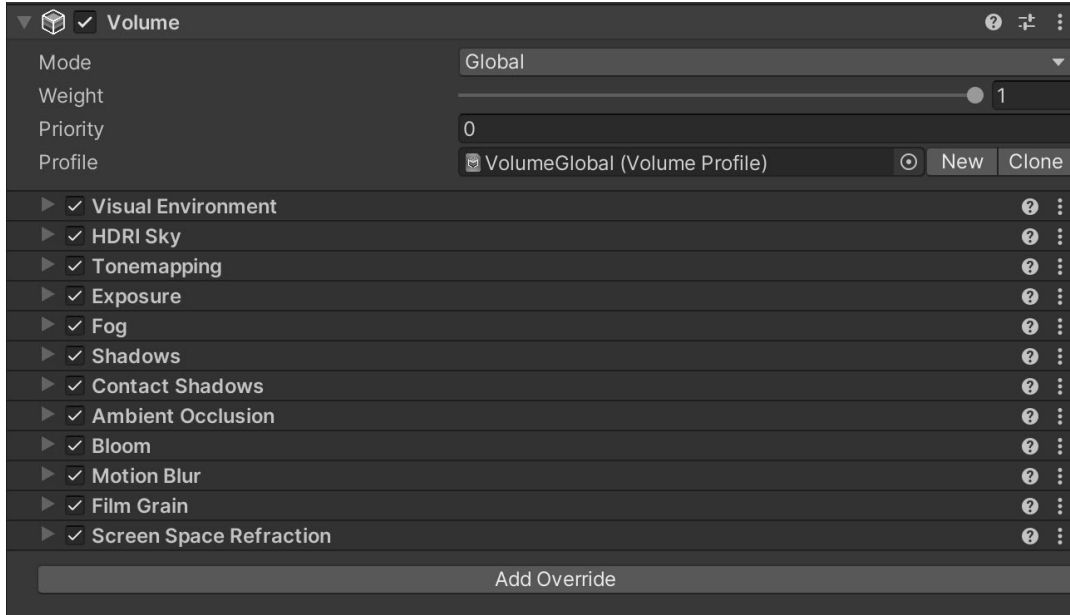


Light Explorer에서 열려 있는 씬의 모든 볼륨 목록을 확인합니다.

로컬 볼륨과 글로벌 볼륨

상황에 따라 Volume 컴포넌트의 **Mode** 설정을 **Global** 또는 **Local**로 설정합니다.

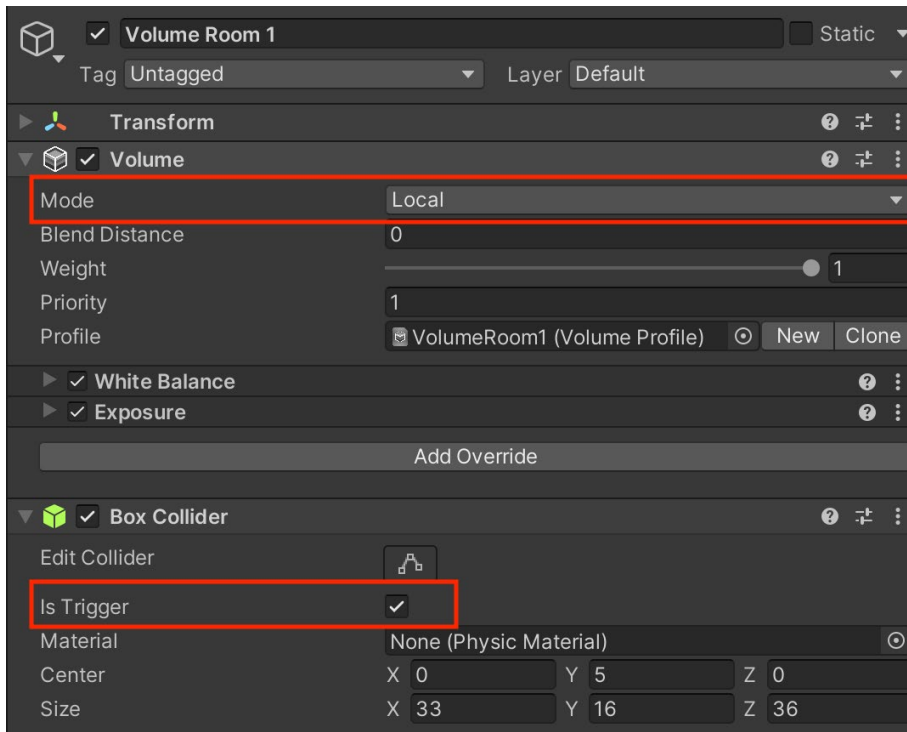
글로벌 볼륨은 경계 없이 포괄적으로 작동하며, 씬의 모든 카메라에 영향을 줍니다. HDRP 템플릿 씬에서 VolumeGlobal은 전체 레벨에 대한 HDRP 설정의 전반적인 기준을 정의합니다.



글로벌 볼륨 오버라이드

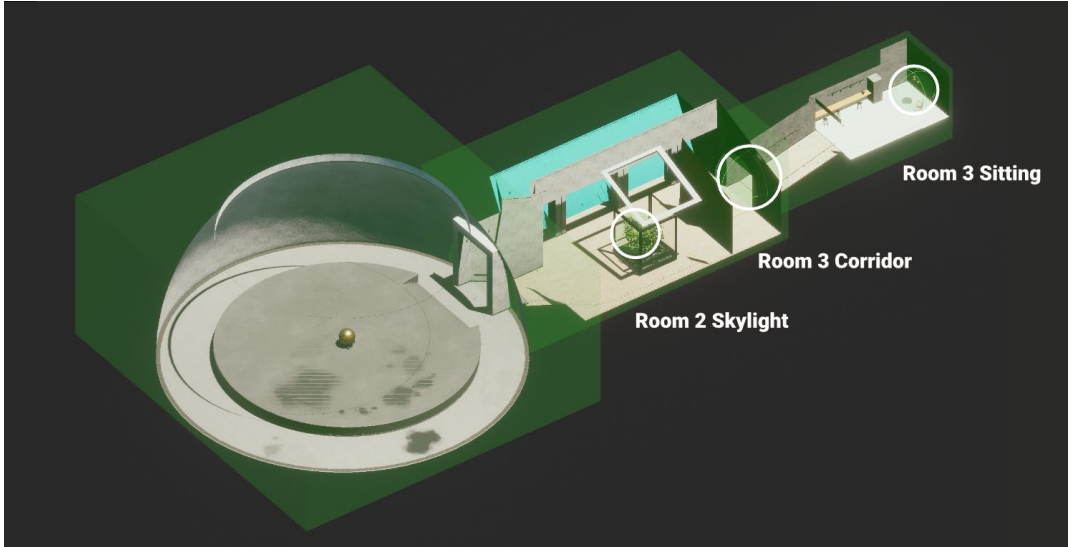
로컬 볼륨은 제한된 공간에 적용되는 설정을 정의하며, 콜라이더 컴포넌트를 사용하여 경계를 정합니다. 콜라이더가 FPS 플레이어 컨트롤러와 같은 물리적 바디의 움직임을 방해하지 않게 하려면 **IsTrigger**를 활성화하세요.

템플릿 씬의 각 공간에는 글로벌 설정을 오버라이드하는 로컬 볼륨이 BoxCollider를 경계로 설정되어 있습니다.



각 공간에는 콜라이더가 IsTrigger로 설정된 로컬 볼륨이 있습니다.

두 번째 공간에는 유리 케이스 옆의 가장 밝은 중심부에 사용되는 작은 구형 볼륨이 있습니다. 마찬가지로, 세 번째 공간에는 입구 복도와 펜던트 조명 아래의 좌석 공간에 더 작은 볼륨이 있습니다.



특별한 조명 조건에는 더 작은 볼륨을 사용합니다.

샘플 씬의 로컬 볼륨은 화이트 밸런스, 노출, 안개 등을 오버라이드합니다. 명시적으로 오버라이드되지 않은 설정은 모두 글로벌 기본값으로 폴백(fallback)됩니다.

카메라가 씬에서 이동할 때 플레이어 컨트롤러가 글로벌 설정을 대신할 로컬 볼륨에 들어가기 전까지는 글로벌 설정이 적용됩니다.

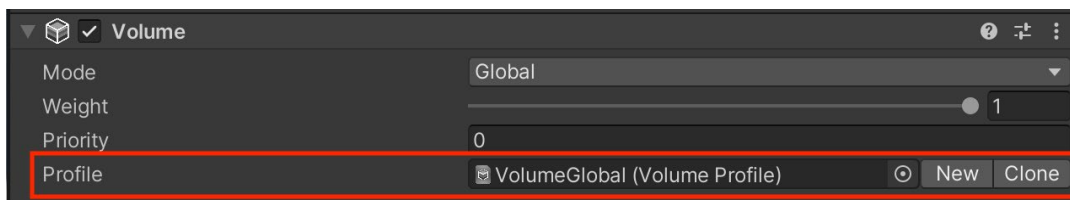


성능 팁

볼륨을 너무 많이 사용하지 않는 것이 좋습니다. 각 볼륨을 평가(블렌딩, 공간화, 오버라이드 계산 등)하려면 CPU 비용이 발생합니다.

볼륨 프로파일

Volume 컴포넌트 자체에는 실제 데이터가 없습니다. 대신 씬을 렌더링하기 위한 HDRP 설정을 포함하고 있는 디스크의 ScriptableObject 에셋인 [Volume Profile](#)을 참조합니다. Profile 필드에서 **New** 버튼이나 **Clone** 버튼을 눌러 새 볼륨 프로파일을 만들 수 있습니다.



Profile 필드를 사용하여 볼륨 프로파일을 전환하거나 새 프로파일을 생성합니다.

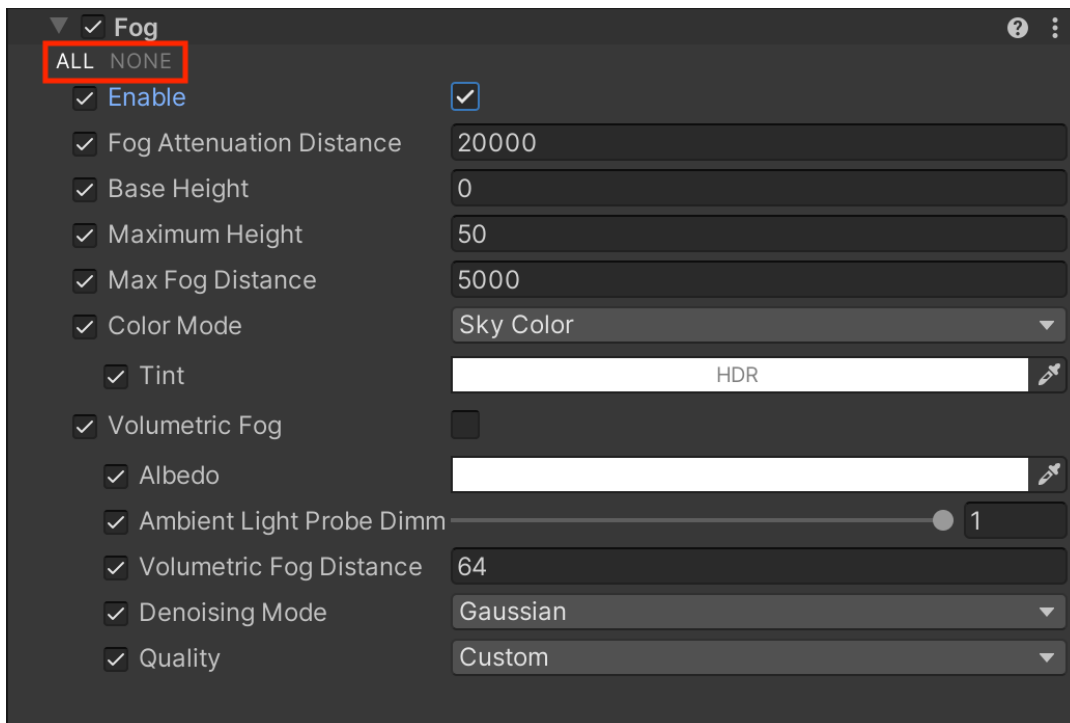
이미 저장되어 있는 다른 프로파일로 전환할 수도 있습니다. 볼륨 프로파일을 파일로 저장하면 이전 설정을 쉽게 재사용하고 볼륨 간에 프로파일을 공유할 수 있습니다.

플레이 모드에서 볼륨 프로파일을 변경한 후 플레이 모드를 종료해도 변경 사항이 사라지지 않습니다.

볼륨 오버라이드

각 볼륨 프로파일은 기본 프로퍼티로 시작합니다. 프로퍼티 값을 편집하려면 볼륨 오버라이드를 통해 개별 설정을 커스터마이징해야 합니다. 예를 들어 볼륨 오버라이드로 볼륨의 안개, 포스트 프로세싱, 노출을 수정할 수 있습니다.

볼륨 프로파일을 설정했다면, **Add Override**를 클릭하여 프로파일 설정을 커스터마이징합니다. Fog 오버라이드를 예로 들면 다음과 같습니다.



볼륨 오버라이드로서의 안개 예시

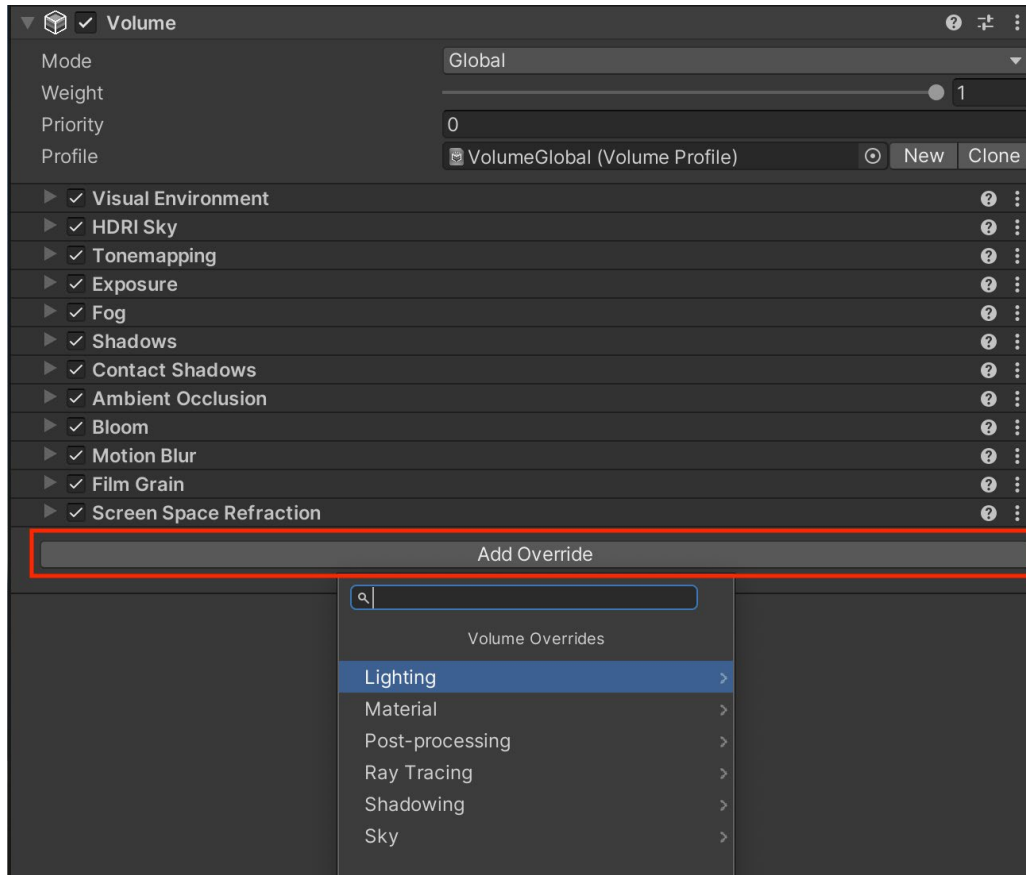
각 볼륨 오버라이드의 프로퍼티에는 왼쪽에 체크박스가 있으며, 이를 사용해 해당 프로퍼티를 편집할 수 있습니다. 체크박스를 비활성화된 상태로 두면 HDRP가 볼륨의 기본값을 사용한다는 의미입니다.

볼륨 오브젝트별로 여러 오버라이드를 적용할 수 있습니다. 오버라이드 항목별로 필요한 프로퍼티를 모두 편집하세요. 왼쪽 상단에 있는 **All** 또는 **None** 슛컷을 사용하여 모든 항목을 빠르게 선택하거나 선택 해제할 수 있습니다.

오버라이드 워크플로

오버라이드 추가는 HDRP의 핵심 워크플로입니다. [프로그래밍에서의 상속](#)과 관련된 개념을 이해하고 있다면, 볼륨 오버라이드도 익숙하게 사용할 수 있을 것입니다.

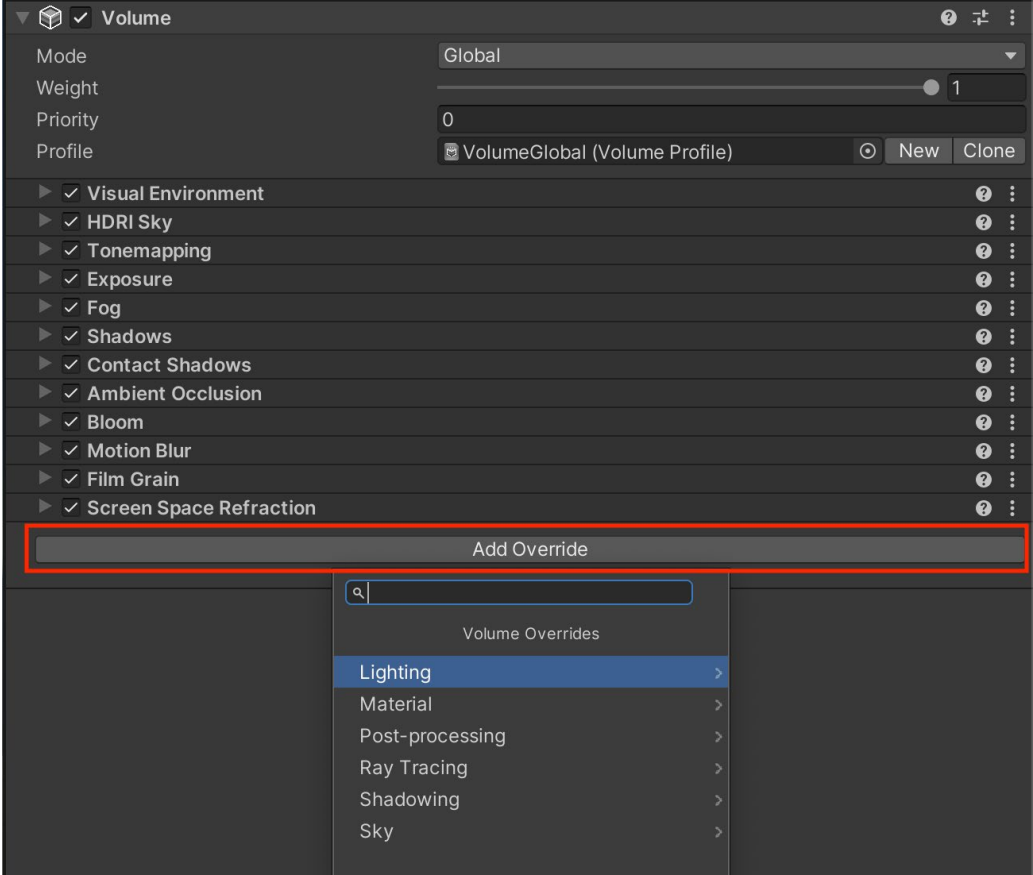
더 높은 수준의 볼륨 설정은 낮은 수준의 볼륨에 대한 기본값이 됩니다. 이 경우 HDRP 기본 설정은 글로벌 볼륨으로 전달되며, 따라서 로컬 볼륨의 기반이 됩니다.



볼륨 오버라이드를 사용한 HDRP 기능 추가

글로벌 볼륨은 HDRP 기본 설정을 오버라이드하고, 로컬 볼륨은 이후 글로벌 볼륨을 오버라이드합니다. **Priority, Weight, Blend Distance** 옵션(아래에서 설명)을 사용하여 겹치는 볼륨으로 인한 충돌을 해결할 수 있습니다.

특정 볼륨 컴포넌트의 현재 값을 디버깅하려면 [Rendering Debugger](#)의 Volume 탭을 사용하세요.



볼륨 디버깅

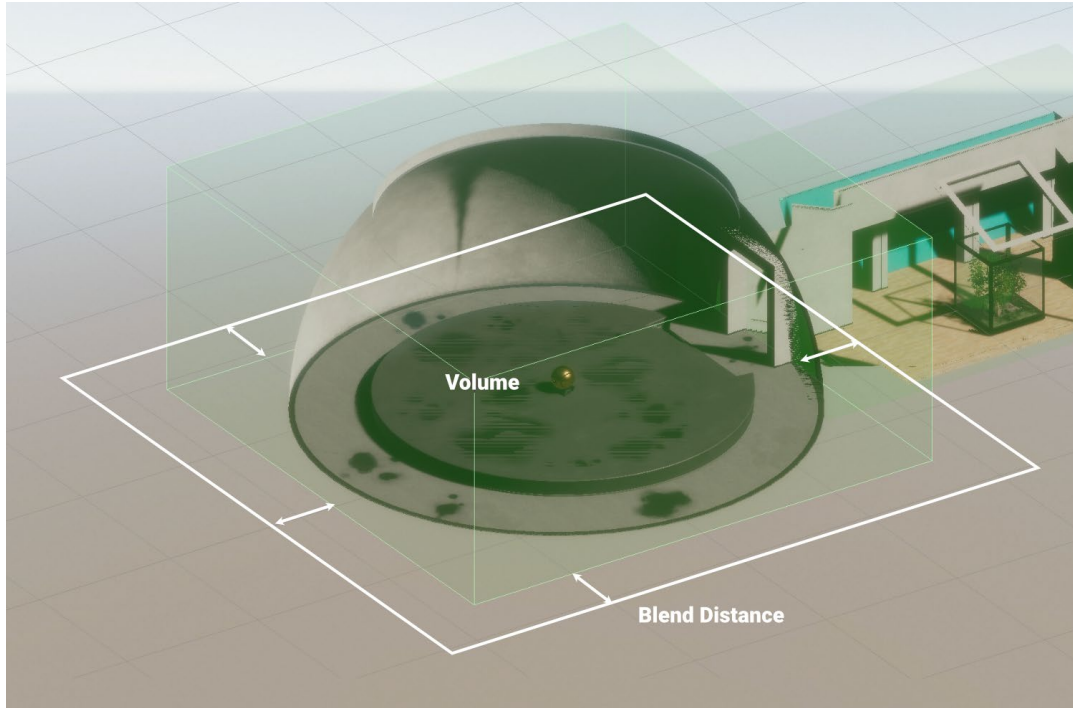
자세한 내용은 HDRP 기술 자료의 [Volume 기술 자료 페이지](#)를 참조하세요.

블렌딩 및 우선순위

레벨마다 둘 이상의 볼륨이 필요한 경우가 많으므로, HDRP를 사용해서 볼륨을 블렌딩할 수 있습니다. 이를 통해 볼륨 사이의 전환을 조금 더 자연스럽게 만들 수 있습니다.

런타임에 HDRP는 카메라 위치를 사용하여 최종 HDRP 설정에 영향을 주는 볼륨을 결정합니다.

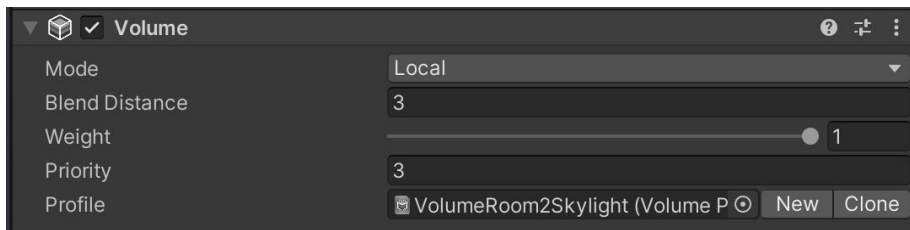
Blend Distance는 볼륨의 콜라이더 외부에서 얼마나 먼 위치부터 페이드가 시작되거나 끝날지 결정합니다. Blend Distance의 값을 0으로 설정하면 볼륨이 즉시 전환된다는 의미이며, 값을 양수로 지정하면 카메라가 지정된 범위에 들어갔을 때부터 볼륨 오버라이드 블렌딩이 시작된다는 의미입니다.



블렌딩 거리는 볼륨 주변의 전환 영역을 정의합니다.

볼륨 프레임워크의 유연성을 활용해 볼륨과 오버라이드를 원하는 대로 적절하게 조합할 수 있습니다. 여러 볼륨이 같은 공간에 겹치는 경우, HDRP는 **Priority** 옵션에 정해진 값에 따라 어떤 볼륨을 우선할지 결정합니다. 설정값이 높을수록 우선순위가 높습니다.

일반적으로 더 정확하게 작업하려면 Priority 값을 명시적으로 설정해야 합니다. 값을 설정하지 않으면 시스템에서 생성 순서를 Priority 값 대신 사용하므로 예상치 않은 결과가 발생할 수 있습니다.



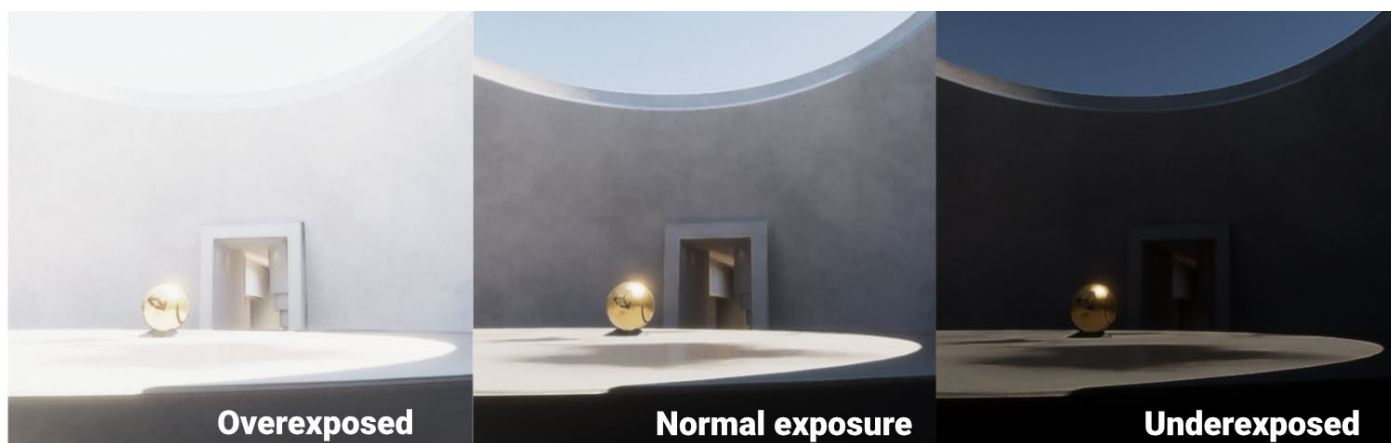
로컬 볼륨을 겹칠 때는 Blend Distance, Weight, Priority를 사용합니다.

노출

HDRP는 실제 조명 모델을 사용하여 씬을 렌더링합니다. 따라서 많은 프로퍼티가 전통적인 사진에서의 설정과 유사합니다.

노출 값의 이해

EV(노출 값)는 카메라의 **셔터 속도**와 **f 값**(렌즈가 열리는 크기 또는 조리개 크기를 결정)의 조합을 나타내는 숫자 값입니다. 이상적인 밝기를 유지하고 어두운 곳과 밝은 곳 모두에서 높은 수준의 디테일을 구현하려면 **노출**을 적절하게 설정해야 합니다. 노출을 적절하게 설정하지 않으면 이미지의 노출 과다나 노출 부족으로 인해 바람직하지 못한 결과를 얻게 됩니다.



노출 과다, 노출 부족, 노출이 적절한 이미지를 비교한 모습입니다.

HDRP의 노출 범위는 일반적으로 다음 범위 내에 있습니다.



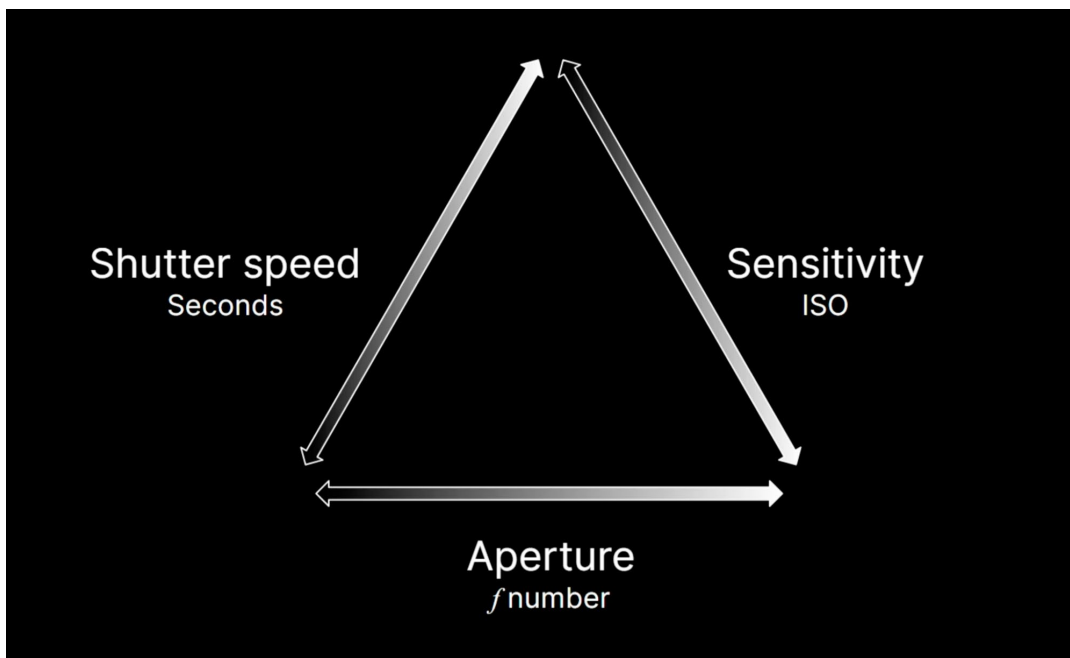
달이 없는 밤부터 밝고 맑은 날까지의 노출 범위

노출 값이 클수록 카메라에 들어오는 광원이 줄어들며, 더 밝은 상황에 적합합니다. 여기서 13에서 16 사이의 EV 값은 화창한 낮 시간대의 실외 환경에 적합합니다. 반대로 어둡고 달이 뜨지 않은 밤하늘의 경우 -3에서 0 사이의 EV 값이 좋습니다.

노출 값을 수정하기 위해 실제 카메라 설정에서 다음과 같은 몇 가지 요소를 변경할 수 있습니다.

- 셔터 속도: 이미지 센서가 빛에 노출되는 시간의 길이
- f 값: 조리개/렌즈가 열리는 크기
- ISO: 필름/센서의 감도

사진 분야에서는 이를 노출의 3요소라고 합니다. Unity에서는 실제 카메라처럼 각 요소의 값 조합을 사용하여 같은 노출 값을 얻을 수 있습니다.



노출의 3요소

HDRP는 모든 노출 값을 EV100으로 표현하며, 이 값의 감도는 100 ISO(국제 표준화 기구) 필름 감도와 같습니다.

노출 값 수식

실제로 노출 값을 계산하는 수식입니다.

$$EV = \log_2 \left(\frac{f \text{ number}^2 / \text{shutter speed}}{ISO / 100} \right)$$

노출 수식

이는 밑수가 2인 로그 스케일입니다. 노출 값이 1단위 증가하면 렌즈에 들어오는 빛의 양이 절반으로 줄어듭니다.

HDRP를 사용하면 실제 이미지와 노출을 맞출 수 있습니다. 카메라나 스마트폰으로 디지털 사진을 촬영하기만 하면 됩니다. 이미지에서 메타데이터를 가져와 f 값, 셔터 속도, ISO를 파악하세요.



디지털 사진의 Exif 데이터로 노출 값을 맞춥니다.

그런 다음 위의 수식을 사용해 노출 값을 계산합니다. 노출 오버라이드(아래 참조)에서 같은 값을 사용하는 경우, 렌더링된 이미지는 실제 이미지 노출과 같을 것입니다.

레벨에 조명을 추가할 때 이런 식으로 디지털 사진을 참고 자료로 사용할 수 있습니다. 반드시 이미지를 완벽하게 재현하는 것이 목표가 아니더라도 실제 사진과 노출 값을 맞추면 조명을 지레짐작하여 설정하지 않아도 됩니다.

노출 오버라이드

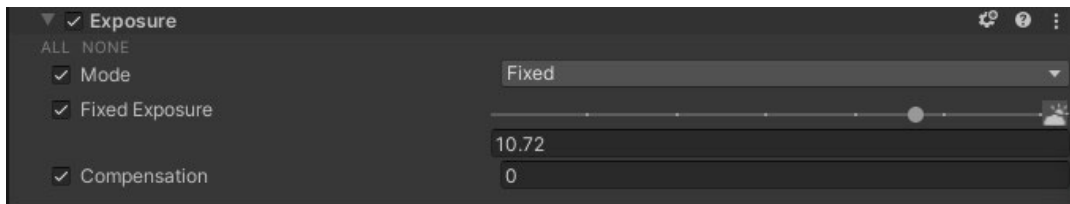
HDRP에서 노출은 볼륨 오버라이드입니다. 어떤 프로퍼티를 사용할 수 있는지 보려면 로컬 볼륨이나 글로벌 볼륨에 노출을 추가하세요.

Mode 드롭다운에서 **Fixed**, **Automatic**, **Automatic Histogram**, **Curve Mapping**, **Physical Camera** 중 한 가지 옵션을 선택할 수 있습니다.

Compensation을 사용하면 노출을 바꾸거나 조정할 수 있으며, 일반적으로 이를 사용하여 노출을 미세하게 조정하거나 렌더링된 이미지에서의 **f 값**을 올리거나 내릴 수 있습니다.

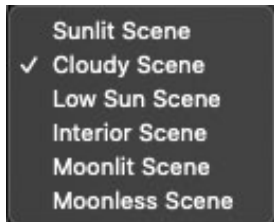
Fixed 모드

Fixed 모드를 사용하면 노출 값을 직접 설정할 수 있습니다.



Fixed 모드 노출

어떤 기능인지 알아보려면 **Fixed Exposure** 슬라이더의 눈금 표시를 따라가 보세요. 오른쪽 아이콘의 드롭다운에는 햇살이 비치는 씬을 위한 13부터 달이 없는 밤하늘 씬을 위한 -2.5까지 여러 프리셋이 마련되어 있습니다. 필드에서 임의의 값을 직접 설정할 수도 있습니다.



Fixed 노출 프리셋

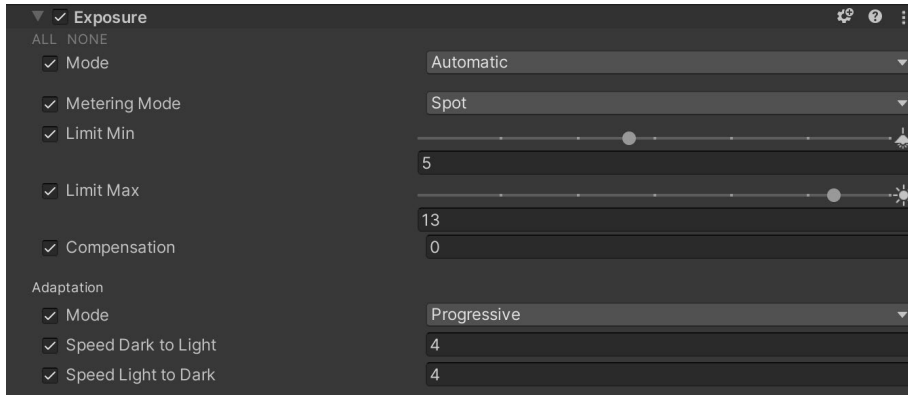
Fixed 모드는 간단하지만, 그리 유연하지는 않습니다. 보통 하나의 노출 값을 전체적으로 적용해 상대적으로 균일한 조명을 사용하는 볼륨이나 씬에서 활용할 만합니다.

Automatic 모드

Automatic 모드는 화면의 밝기 수준 범위에 따라 노출을 동적으로 설정합니다. 이는 **인간의 눈이 다양한 어둠기에 암순응**하여 검은색으로 인식하는 정도를 재정의하는 것과 매우 비슷합니다.

Automatic 모드는 다양한 조명 상황에서 사용할 만하지만, 카메라가 씬에서 매우 밝거나 매우 어두운 부분을 바라보는 경우 의도치 않게 이미지의 노출이 과다하거나 부족할 수 있습니다.

Limit Min과 **Limit Max**를 사용하여 노출 수준을 원하는 범위 내로 유지하세요. 플레이 테스트를 통해 작업자가 제한한 범위가 레벨 전반에서 예상하는 노출 범위 내에 있는지 확인하는 것이 좋습니다.



Automatic 모드 노출

Metering Mode는 마스크 옵션과 함께 사용하여 자동 노출에 사용할 프레임 부분을 정할 수 있습니다.

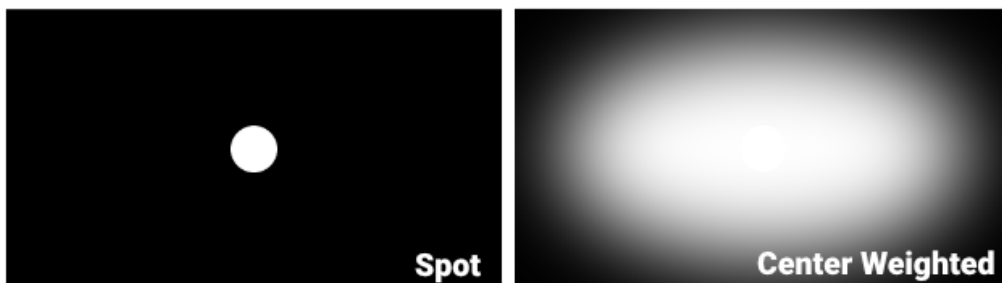
Adaptation Mode는 속도를 조정하는 옵션을 통해 카메라가 어두운 곳에서 밝은 곳으로, 또는 그 반대로 전환될 때 자동 노출이 변경되는 방식을 제어합니다. 인간의 눈처럼 카메라를 매우 어두운 곳에서 매우 밝은 곳으로 옮기거나 그 반대로 옮기게 되면 잠시 혼란이 발생할 수 있습니다.

Metering Mode 옵션

Automatic, Automatic Histogram, Curve Mapping 모드에서는 Metering Mode를 사용하여 노출을 계산할 때 어떤 프레임 부분을 사용할지 제어합니다. Metering Mode를 다음과 같이 설정할 수 있습니다.

Average: 카메라가 전체 프레임을 사용하여 노출을 측정합니다.

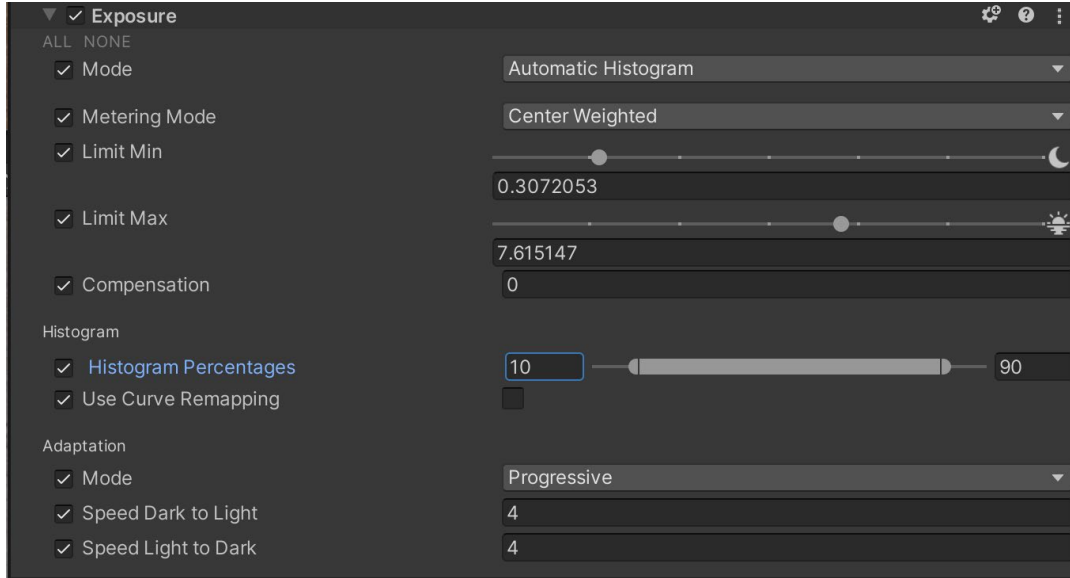
- **Spot:** 카메라가 화면 중앙 부분만 사용하여 노출을 측정합니다.
- **Center Weighted:** 카메라가 이미지 중앙의 픽셀을 중점적으로 사용하고 프레임 모서리 쪽으로 갈수록 점차 사용하지 않습니다.
- **Mask Weighted:** 제공된 이미지(Weight Texture Mask)가 노출을 결정할 때 어떤 픽셀이 가장 중요한지 정합니다.
- **Procedural Mask:** 카메라가 절차적으로 생성된 텍스처를 기반으로 노출을 측정합니다. 중심, 반지름, 매끄러운 정도에 대한 옵션을 변경할 수 있습니다.



Spot 모드와 Center Weighted Metering Mode

Automatic Histogram

Automatic Histogram 모드는 Automatic 모드를 한 단계 더 발전시킨 모드로, 이미지에 대한 [히스토그램](#)을 계산하고 노출을 설정할 때 가장 어둡고 가장 밝은 픽셀은 무시합니다.

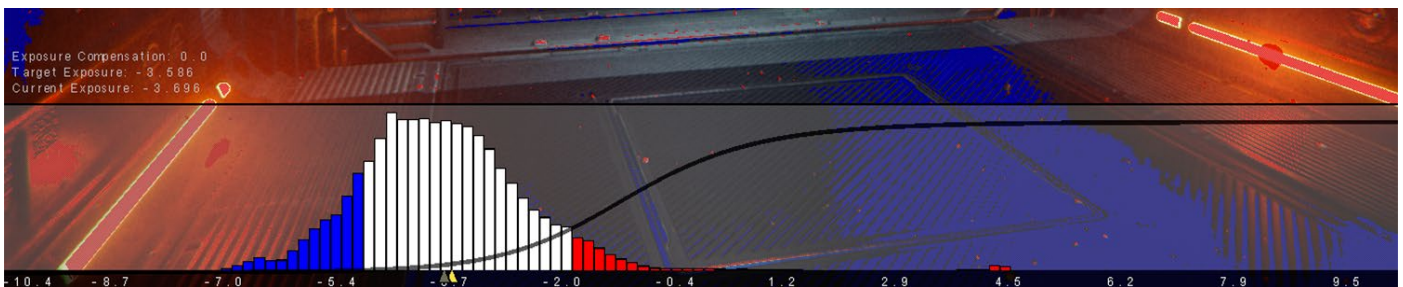


Automatic Histogram 모드

노출을 계산할 때 가장 어둡거나 가장 밝은 픽셀을 무시하면 프레임에 매우 밝거나 어두운 픽셀이 나타날 때마다 더 안정적인 노출을 얻을 수 있습니다. 이러한 방법으로 강렬한 발광 표면이나 검은색 머티리얼로 인해 렌더링된 결과물의 노출이 지나치게 부족하거나 과하지 않도록 조절할 수 있습니다.

Histogram Percentages 설정을 사용하면 히스토그램에서 주어진 백분율 범위를 벗어나는 모든 것을 무시할 수 있습니다. 히스토그램의 가장 왼쪽 및 오른쪽 부분에서 가장 밝은 픽셀과 가장 어두운 픽셀을 잘라낸다고 생각하면 됩니다.

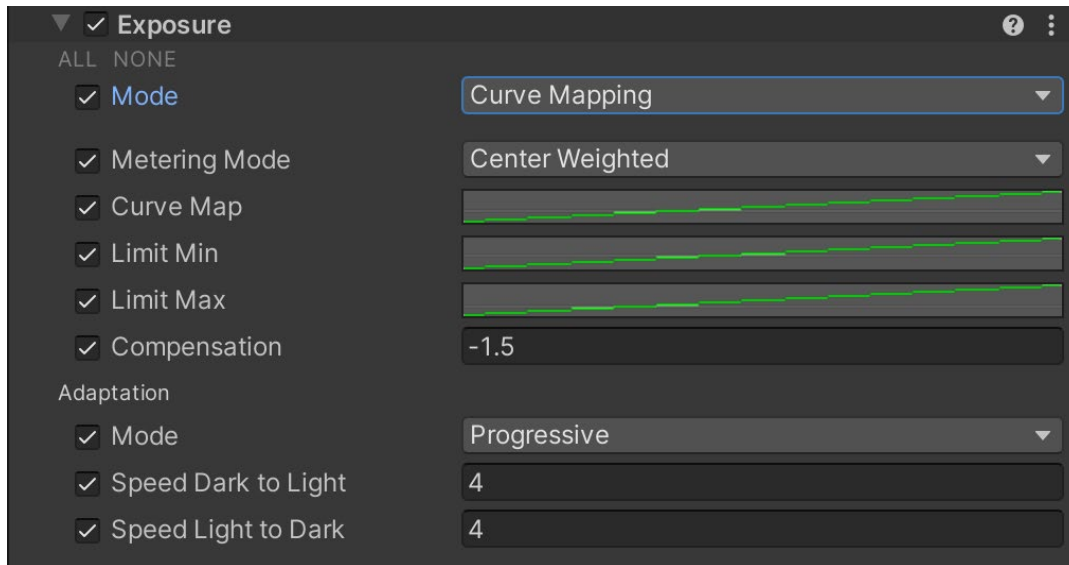
Curve Remapping을 사용하면 노출 커브 역시 다시 매핑할 수 있습니다(아래의 Curve Mapping 참조).



Automatic Histogram 모드에서는 히스토그램中间的 픽셀을 사용해 노출을 계산합니다.

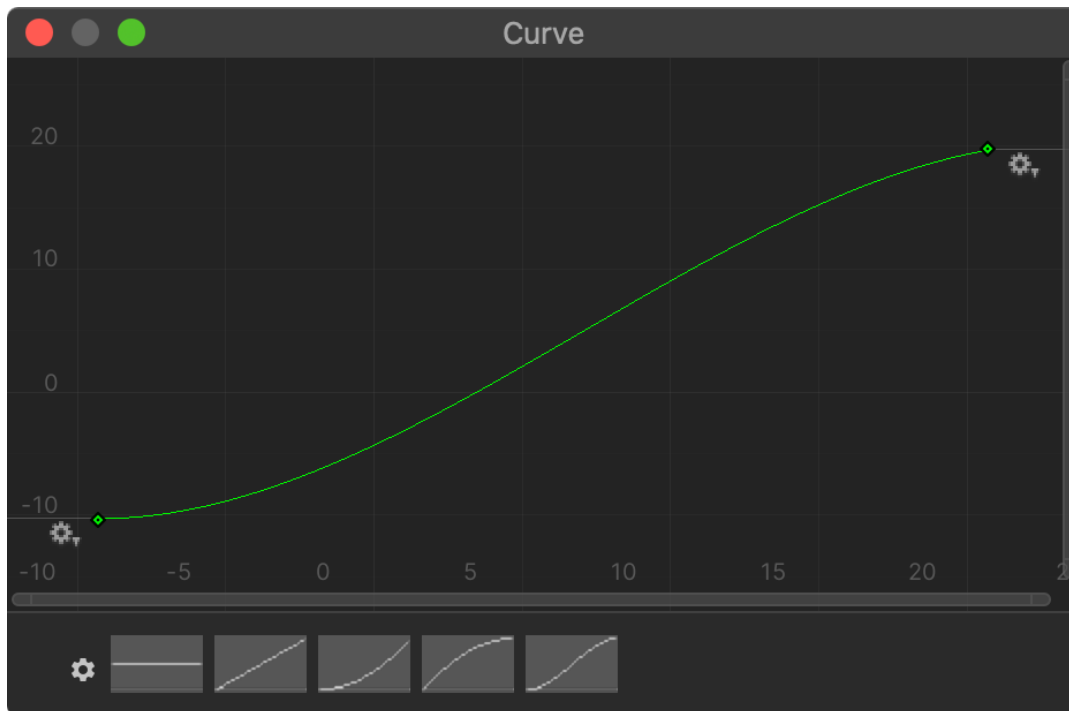
Curve Mapping

Curve Mapping 모드는 [Automatic](#) 모드의 또 다른 버전입니다.



Curve Mapping 모드

여기서 커브의 x축은 현재 노출을 나타내고 y축은 타겟 노출을 나타냅니다. 노출 커브를 다시 매핑하면 매우 정확한 결과를 얻을 수 있습니다.

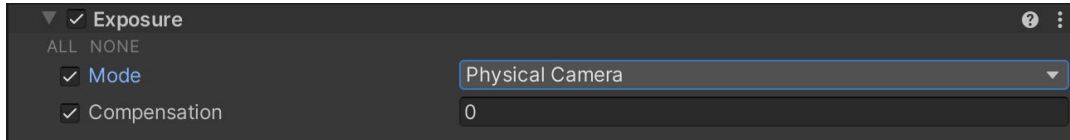


커브를 사용해 노출을 조정합니다.

물리적 카메라

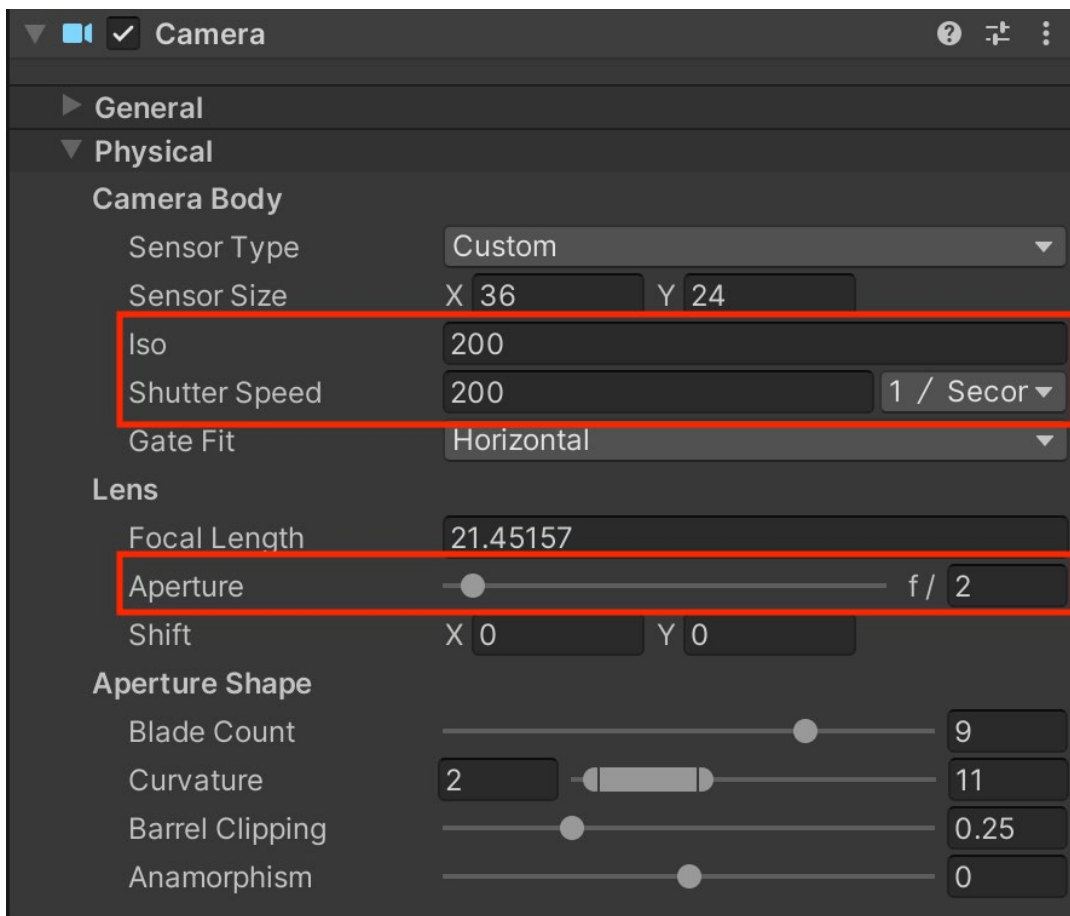
사진 촬영에 익숙하다면 **Physical Camera** 모드가 카메라 파라미터를 설정하는 데 도움이 될 수 있습니다.

Exposure 오버라이드의 **Mode**를 **Physical Camera**로 전환한 다음 Main Camera를 찾습니다.



Physical Camera 모드

Physical Camera를 활성화하면, 인스펙터에 다음 프로퍼티가 나타납니다.



카메라의 물리적 카메라 프로퍼티

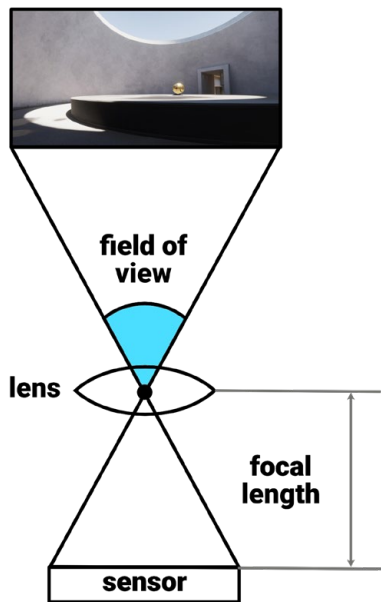
노출에서 중요한 요소는 **ISO**(감도), **조리개**(또는 f 값), **셔터 속도**입니다. 레퍼런스 사진과 값을 맞추려는 경우, 이미지의 **Exif** 데이터에서 올바른 설정을 복사하면 됩니다. 아니면 **이 표**를 활용하여 f 값과 셔터 속도를 기반으로 노출 값을 추측할 수 있습니다.

더 많은 물리적 카메라 파라미터

노출과 관련된 것은 아니지만, 다른 **물리적 카메라** 프로퍼티도 실제 카메라 속성에 맞추는 데 도움이 될 수 있습니다.

예를 들어 Unity와 기타 다양한 3D 애플리케이션에서는 일반적으로 **시야각(FOV)**을 사용하여 카메라가 한 번에 볼 수 있는 월드의 범위를 결정합니다.

그러나 실제 카메라의 **시야각**은 센서의 크기와 렌즈의 초점 거리에 따라 달라집니다. 직접 시야각을 설정하는 대신 물리적 카메라 설정을 사용하면 실제 카메라 데이터에서 **Sensor Type**, **Sensor Size**, **Focal Length** 값을 가져올 수 있습니다. 그러면 Unity가 해당 Field of View 값을 자동으로 계산합니다.



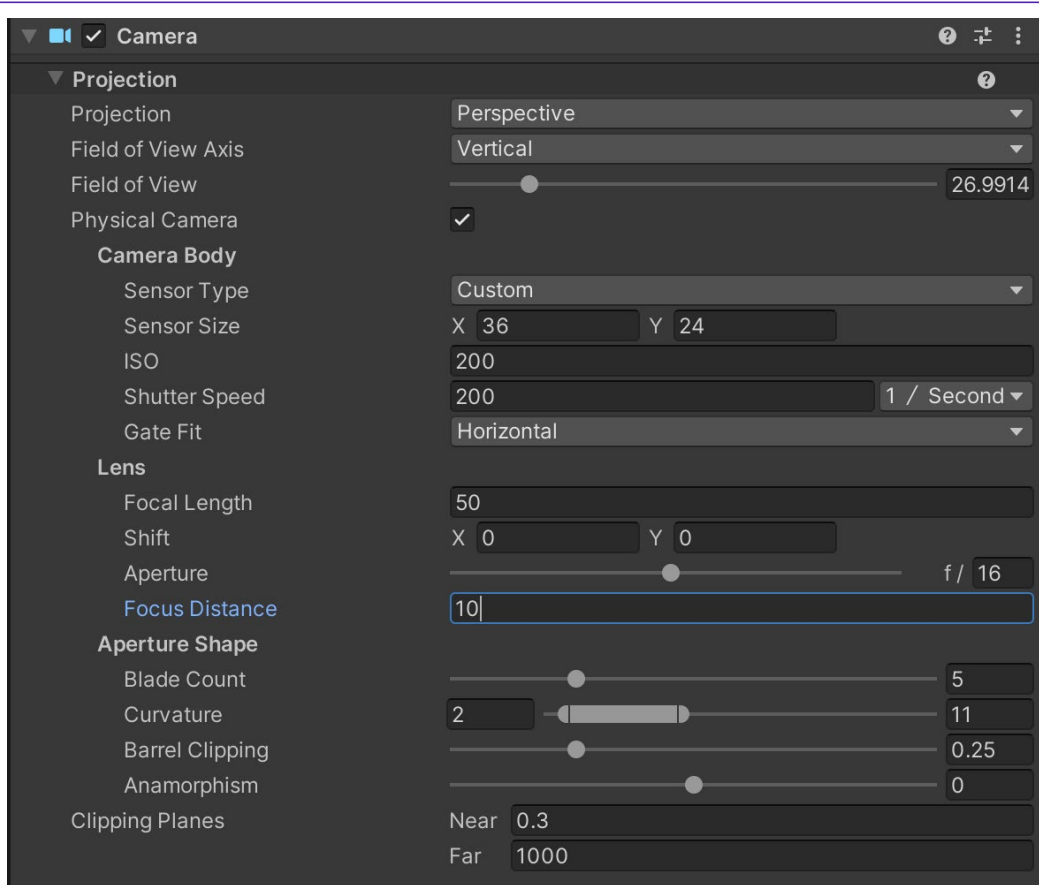
초점 거리, 센서 크기, 시야각의 관계

실제 사진 레퍼런스와 일치하게 만들려면 이미지 파일에 포함된 카메라 메타데이터를 활용하세요. Windows와 macOS 모두 디지털 이미지로부터 Exif 데이터를 읽을 수 있습니다. 해당 필드의 값을 가상 카메라에 복사하면 됩니다.

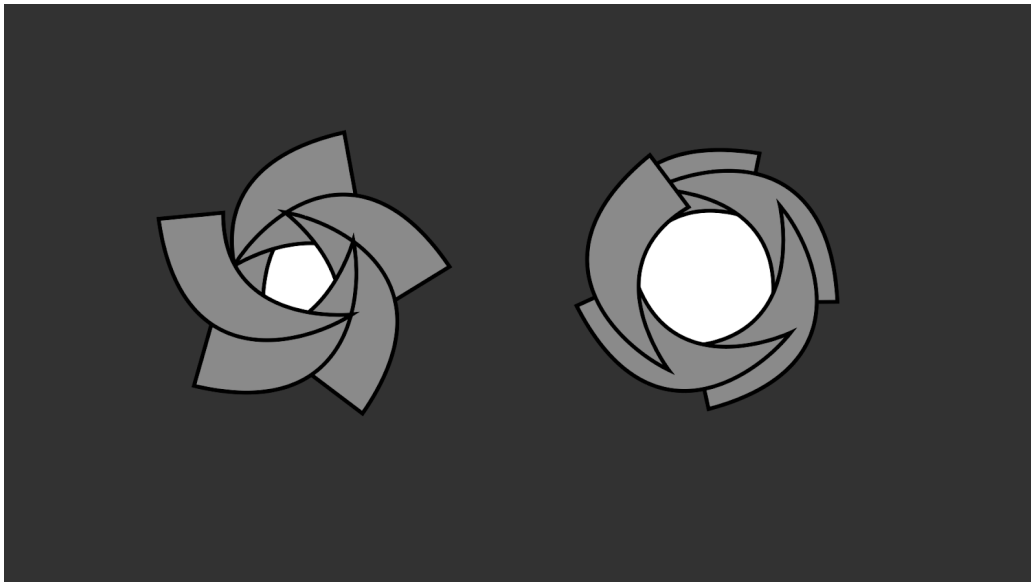
참고: 메타데이터에서 카메라 제조업체와 모델을 확인한 후, 제조업체의 웹사이트에서 정확한 센서 수치를 검색해야 할 수도 있습니다. [이 문서](#)에서 일반적인 이미지 센서 형식의 추정 수치를 확인할 수 있습니다.

아래에 있는 파라미터 중 일부는 Depth of Field Volume에 영향을 줍니다.

카메라의 인스펙터에서 **Focus Distance**를 제어할 수 있습니다. Depth of Field Volume 컴포넌트에서 **Focus Mode**와 **Focus Distance Mode**를 **Physical Camera**로 설정합니다.



Blade Count, Curvature, Barrel Clipping은 카메라의 조리개 모양을 바꿉니다. 이는 [Depth of Field](#) 볼륨 컴포넌트로 인해 생기는 보케(bokeh)의 모양에 영향을 줍니다.



물리적 카메라 파라미터를 사용해 조리개 모양을 변경할 수 있습니다. 조리개날이 5개인 조리개의 경우 오각형 보케(왼쪽) 또는 원형 보케(오른쪽)를 생성할 수 있습니다.

광원

HDRP에는 씬의 조명 제어를 지원하는 다양한 광원 유형과 광원 셰이프가 포함되어 있습니다.

광원 유형

Unity의 다른 렌더 파이프라인과 마찬가지로 다음과 같은 광원 유형을 사용할 수 있습니다.

- **방향 광원:** 강도가 줄어들지 않는 완벽하게 평행한 광선으로 무한히 먼 곳에서 오는 광원처럼 작동합니다. 방향 광원은 보통 태양 빛을 표현합니다. 실외 씬에서는 주로 방향 광원을 주요 광원으로 사용합니다.
- **스팟 광원:** 현실의 스포트라이트와 유사하며 원뿔형, 피라미드형, 박스형이 있습니다. 스포트 광원은 원뿔형이나 피라미드형의 모서리 방향으로, 그리고 z축을 따라 감쇠합니다.
- **점 광원:** 점 광원은 전방위 광원으로 공간의 한 지점에서 모든 방향을 비춥니다. 등불이나 양초 등의 복사 광원에 활용할 수 있습니다.
- **면 광원:** 직사각형, 원통형, 원판형 등 특정 모양의 표면에서 빛을 투사합니다. 면 광원은 창이나 형광등과 같이 중앙의 강도가 균일한 넓은 광원처럼 작동합니다.

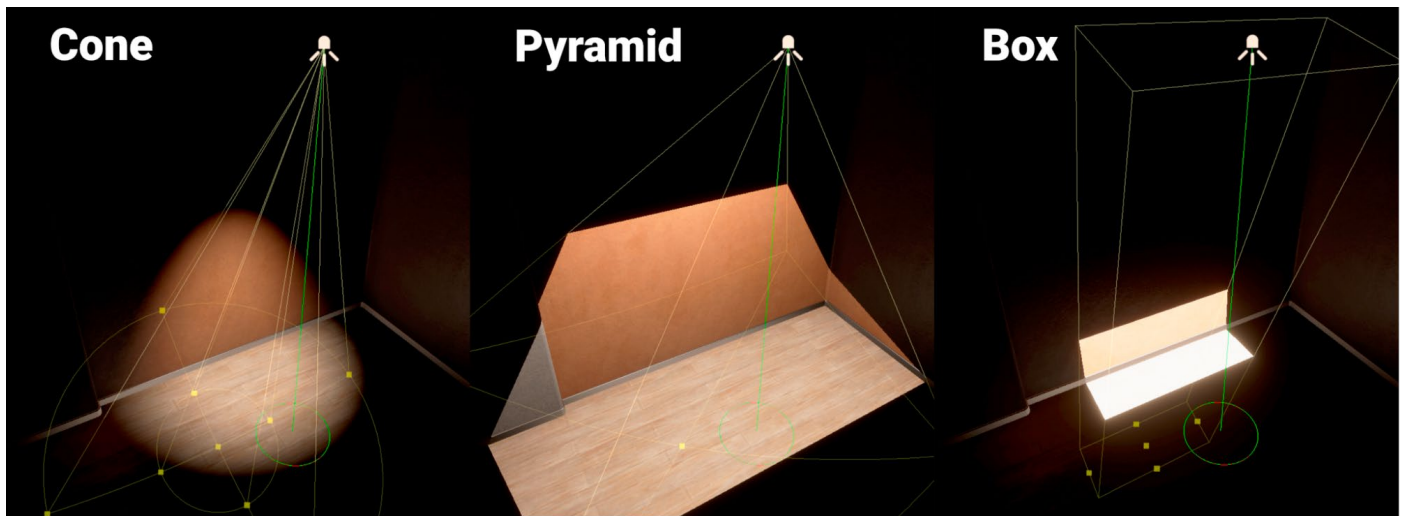
Range를 사용하여 스포트 광원, 점 광원, 면 광원이 감쇠하는 방식을 수정할 수 있습니다. 대부분의 HDRP 광원은 실제 광원처럼 **역제곱 법칙**으로 감소됩니다.

셰이프

스팟 광원과 면 광원에는 각 광원이 감쇠하는 방식을 제어할 수 있는 추가 셰이프가 있습니다.

HDRP 스포트 광원은 세 가지 셰이프를 사용할 수 있습니다.

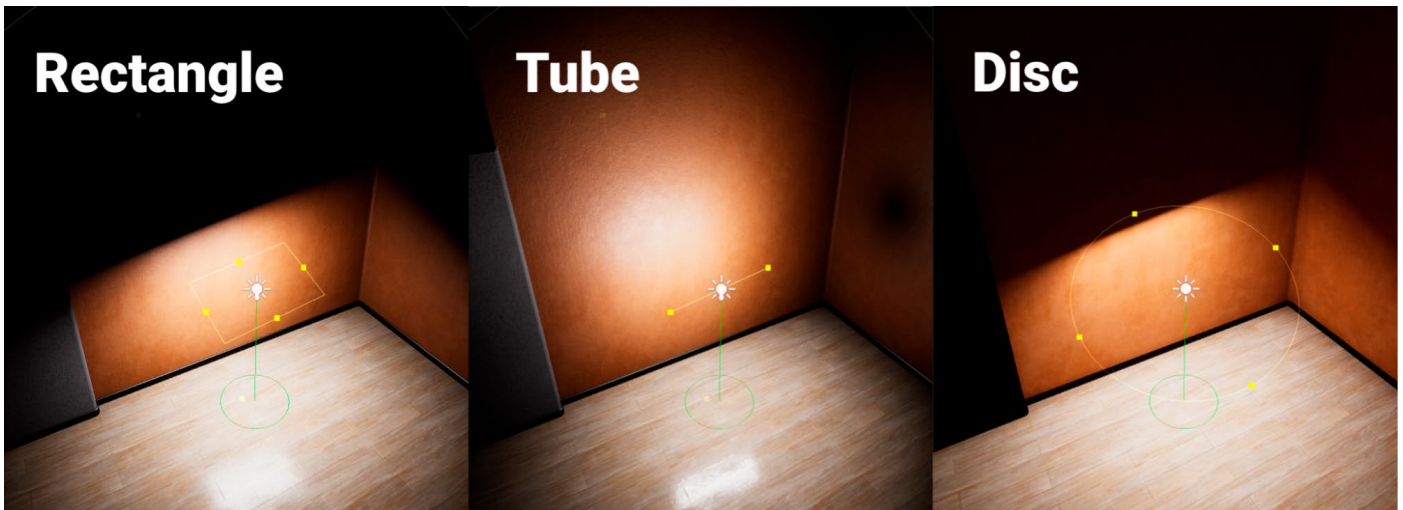
- **원뿔**: 하나의 지점에서 원형 바닥으로 광원을 투사합니다. **Outer Angle(°)**과 **Inner Angle(%)** 값을 조정하여 원뿔 모양을 잡고 각도 감쇠를 수정할 수 있습니다.
- **피라미드**: 하나의 지점에서 사각형 바닥으로 광원을 투사합니다. **Spot Angle**과 **Aspect Ratio** 값을 조정하여 피라미드 모양을 잡을 수 있습니다.
- **박스**: 직사각형 영역에 광원을 균일하게 투사합니다. X 및 Y 크기는 기본 직사각형 크기를 정하고 Range는 Y 크기를 제어합니다. 박스형 광원은 **Range Attenuation**이 선택되어 있지 않는 한 감쇠가 발생하지 않으며, 박스 경계 내의 햇빛을 시뮬레이션할 때 사용할 수 있습니다.



HDRP 스포트 광원 셰이프

HDRP 면 광원은 세 가지 셰이프를 사용할 수 있습니다.

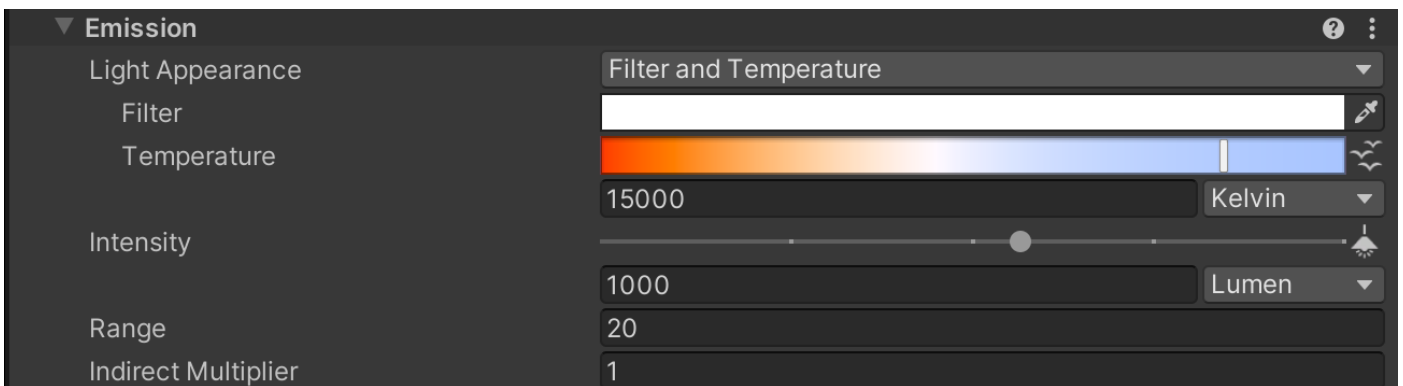
- **직사각형**: 정의된 범위까지 양의 로컬 Z 방향으로 직사각형 셰이프의 광원을 투사합니다.
- **원통형**: 정의된 범위까지 모든 방향으로 한 줄로 이뤄진 광원을 투사합니다. 이 광원은 Realtime 모드에서만 작동합니다.
- **원판형**: 정의된 범위까지 양의 로컬 Z 방향으로 원판 셰이프의 광원을 투사합니다. 이 광원은 Baked 모드에서만 작동합니다.



HDRP 면 광원 셰이프

색상 및 온도

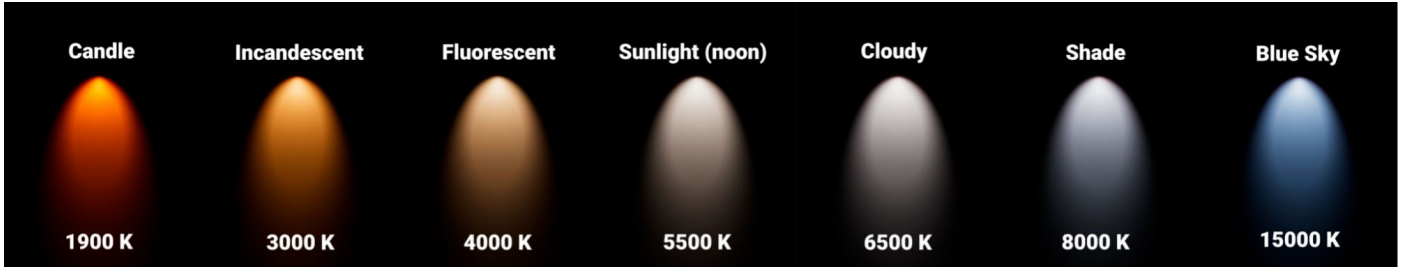
모든 HDRP 광원 유형에는 광원의 모습을 정의하는 **Emission** 프로퍼티가 있습니다.



Emission에서 Light Appearance 프로퍼티를 수정합니다.

Light Appearance를 **Color**로 바꾸고 **RGB 색상**을 지정할 수 있습니다. 아니면 **Filter and Temperature**로 바꾸고 물리적으로 더 정확하게 입력할 수 있습니다.

색 온도는 **켈빈도**를 기준으로 색상을 설정합니다. 조명 및 노출 요약 자료를 참조하세요.



켈빈 스케일의 색 온도

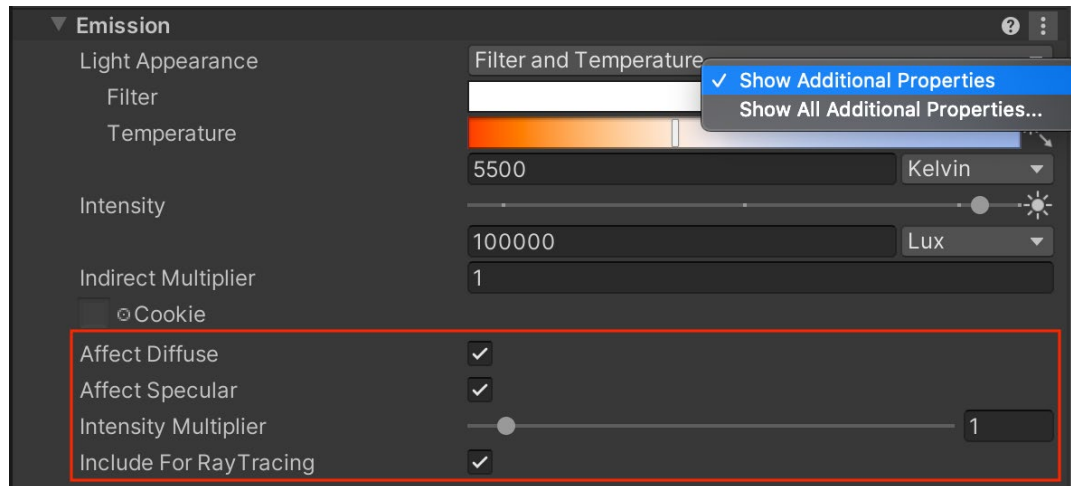
필터와 유사한 역할을 하는 다른 색상을 추가하면 광원에 다른 색조를 적용할 수도 있습니다. 사진 촬영 시 **색상 젤 필터**를 추가하는 것과 유사합니다.

더 많은 프로퍼티

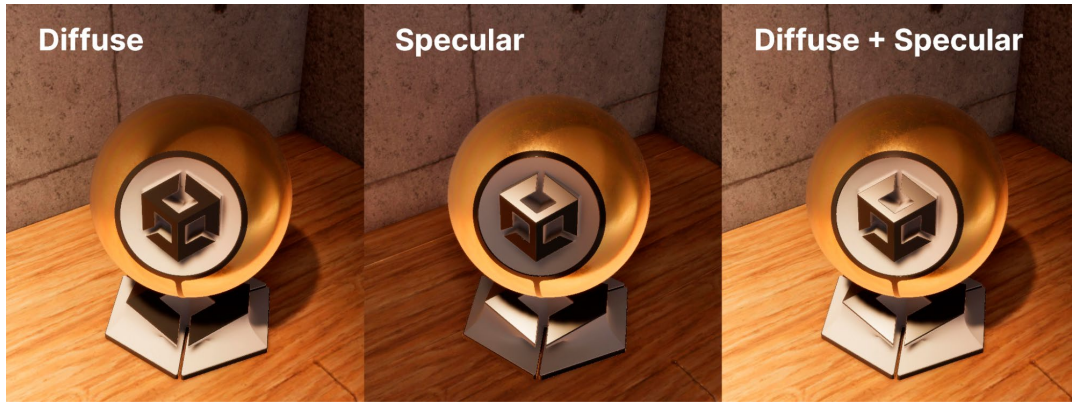
HDRP에는 인스펙터 프로퍼티의 오른쪽 상단에 있는 **더 보기 메뉴()** 아래에 더 많은 고급 제어 프로퍼티가 있습니다. 더 많은 옵션을 확인하려면 **Show Additional Properties**를 선택합니다.

Affect Diffuse와 **Affect Specular**를 토글하는 기능도 있습니다. 예를 들어 컷씬이나 시네마틱 조명에서 밝고 반짝이는 하이라이트를 제어하는 광원과 부드러운 산란광을 만들어내는 광원을 독립적으로 분리할 수 있습니다.

또한 **Intensity Multiplier**를 사용하면 원래 강도 값을 변경하지 않고도 광원의 전반적인 강도를 조정할 수 있습니다. 이 옵션은 여러 광원을 한 번에 밝게 하거나 어둡게 할 때 유용합니다.



광원별로 추가 프로퍼티를 활용할 수 있습니다.

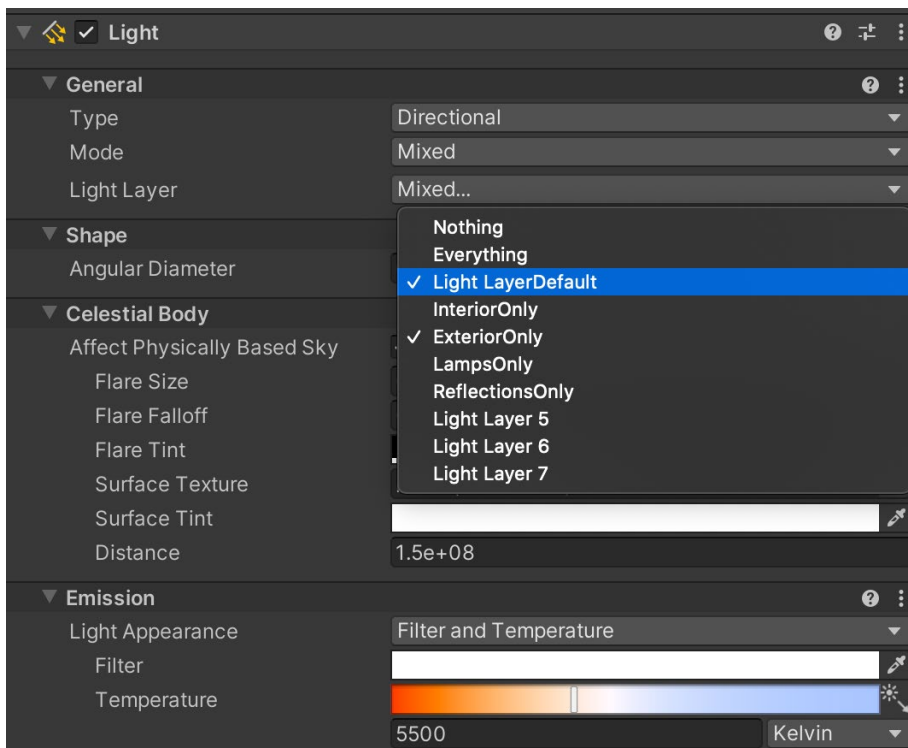


산란광과 스페큘러 조명을 토글하여 추가로 제어합니다.

광원 레이어

HDRP에서는 **광원 레이어**를 사용하여 광원이 씬의 특정 메시에만 영향을 주도록 할 수 있습니다. 광원 레이어는 Light 컴포넌트 및 MeshRenderer와 연결할 수 있는 LayerMask입니다.

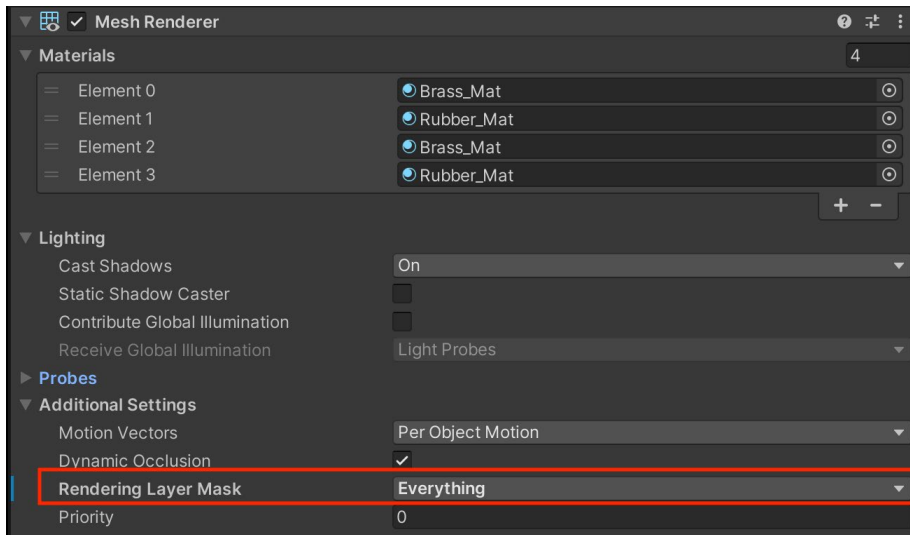
Light 프로퍼티에서 **More Options** 버튼()을 클릭하면, **General** 아래에 **Light Layer** 드롭다운이 표시됩니다. 여기서 광원과 함께 사용할 Layer Mask를 선택합니다.



광원 레이어를 선택합니다.

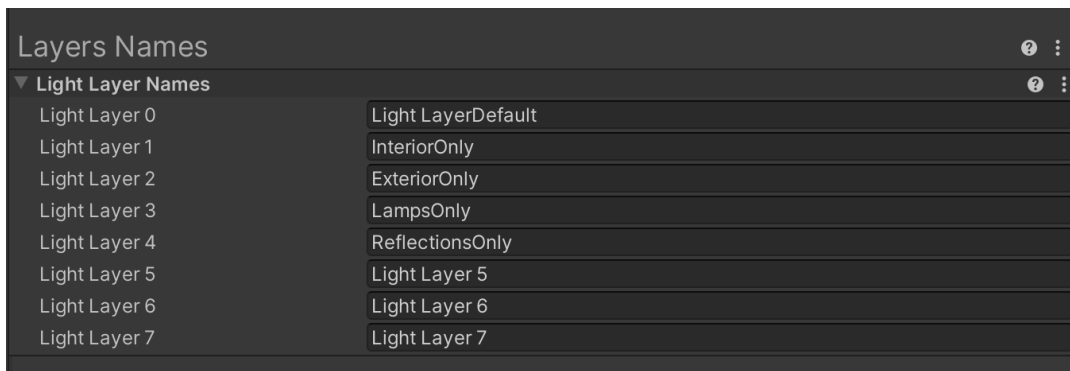
그런 다음, **Rendering Layer Mask**에서 MeshRenderer를 설정합니다. 일치하는 LayerMask의 광원만 메시에 영향을 줍니다. 이 기능은 빛 번짐 현상을 수정하여 광원이 의도한 대상에만 비치도록 하는 데 매우 유용합니다. 또한 캐릭터가 전용 시네마틱 조명만 받도록 컷씬 조명을 설정하는 워크플로에 활용할 수도 있습니다.

예를 들어 건물 내부의 조명이 실수로 벽을 관통해 건물 외부를 비추는 것을 방지하려면, 건물의 내부와 외부에 대한 명확한 광원 레이어를 설정하면 됩니다. 그러면 광원 설정을 세부적으로 제어할 수 있습니다.



Rendering Layer Mask를 설정해 특정한 광원만 메시에 영향을 미치게 만들 수 있습니다.

광원 레이어를 설정하려면 **HDRP Default Settings**로 이동합니다. **Layers Names** 섹션에서 **Light Layer 0~7**의 문자열 이름을 지정할 수 있습니다.



HDRP Default Settings의 Layers Names 섹션

전체 Light 프로퍼티 목록을 비롯한 더 자세한 내용은 [Light 컴포넌트 기술 자료](#)를 참조하시기 바랍니다.

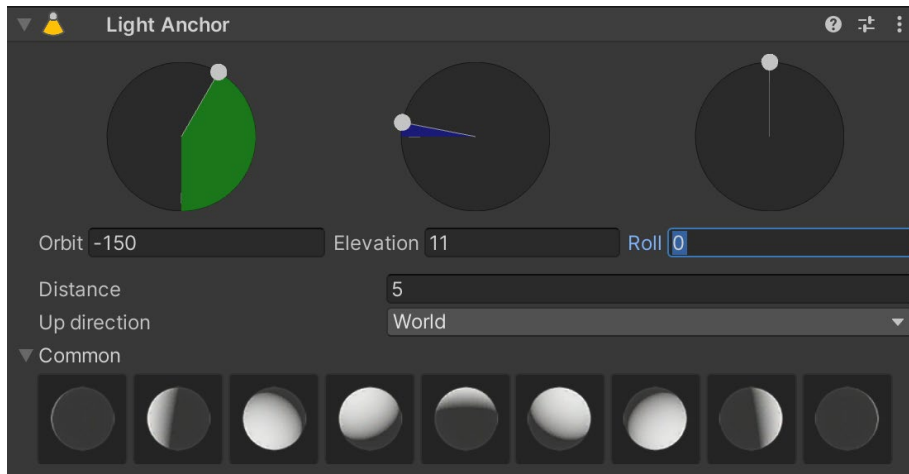
광원 앵커

Unity 6에서는 **광원 앵커** 시스템이 있어서 카메라와 피사체 사이의 각도와 거리를 조정해 광원을 빠르게 설정할 수 있습니다. 또한 9가지 프리셋을 통해 일반적인 조명 각도를 선택할 수 있습니다.

캐릭터나 프랍(prop) 주변에 여러 광원을 사용하는 시네마틱에서 씬, 제품, 장면에서 광원이 필요한 경우 Light Anchor 컴포넌트를 사용하여 앵커 대상 주변의 스크린 공간에서 빠르게 광원을 조작할 수 있습니다.

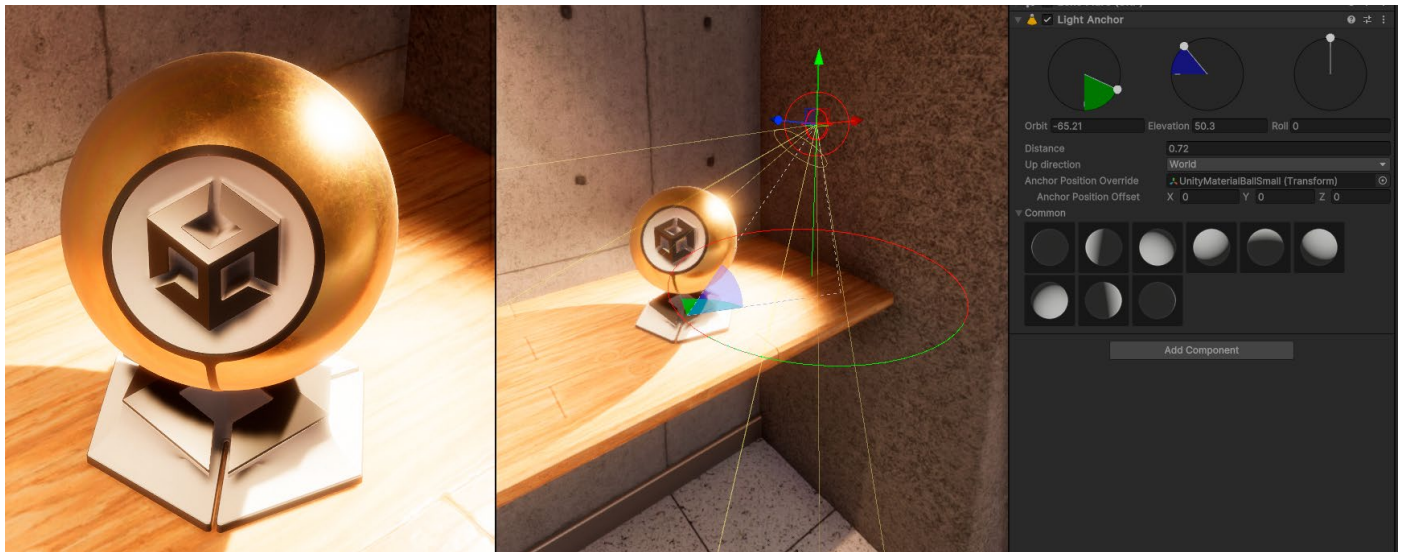
광원 앵커를 사용하려면 먼저 카메라에 MainCamera 태그를 지정해야 합니다. 그런 다음 제어하려는 스포트 광원에 **Light Anchor** 컴포넌트를 추가합니다. 그리고 피사체에 광원을 정렬합니다. 피사체의 위치는 이제 스포트 광원의 앵커 포인트가 됩니다. 앵커 포인트와 스포트 광원 사이의 **Distance** 값을 늘립니다.

이제 씬(Scene) 뷰에서 광원의 Transform을 직접 조정하지 않고도 게임(Game) 뷰 내에서 광원의 **Orbit**, **Elevation**, **Roll**을 조정하여 앵커 포인트 주변의 광원 위치를 조정할 수 있습니다.



Light Anchor 컴포넌트

더 자세한 내용은 [이 프레젠테이션](#)에서 광원 앵커를 소개하는 부분을 참고하세요.



트랜스폼을 직접 변경하는 대신 Light Anchor 컴포넌트로 광원 방향을 조정합니다.

물리 광원 단위 및 강도

HDRP는 PLU(물리 광원 단위)를 사용하여 빛 강도를 측정합니다. PLU는 칸델라, 루멘, 럭스, 니트를 비롯한 조도의 실제 SI 측정값과 일치합니다. PLU는 정확도를 위해 Unity에서의 1유닛이 1미터와 동일하다고 예상합니다.

단위

PLU는 광속 단위와 조도 단위를 모두 포함할 수 있습니다. 광속은 광원에서 방출되는 빛의 총량을 나타내며, 조도는 오브젝트가 받는 광원의 총량(보통 단위 면적당 광속)을 나타냅니다.

상업용 조명과 사진은 적용 분야에 따라 단위를 다르게 표현할 수 있으므로 Unity는 호환성을 위해 여러 PLU를 지원합니다.

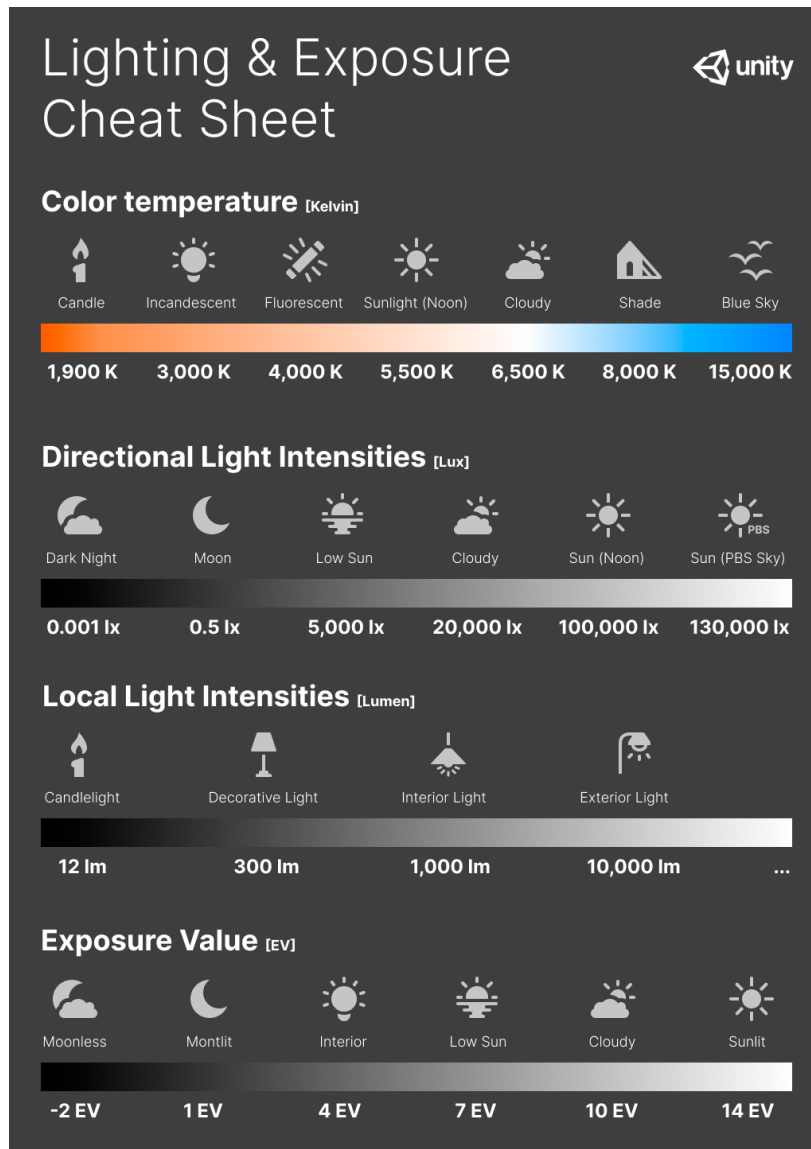
칸델라: 1칸델라는 양초 하나의 광속과 같으며, 촉광이라고도 합니다.

- **루멘:** 광속의 SI 단위로, 입체각(스테라디안)에 대한 1칸델라로 정의됩니다. 보통 시중에 판매되는 전구 사양에서 루멘 정보를 볼 수 있습니다. 루멘은 Unity에서 스폷 광원, 점 광원, 면 광원에 사용합니다.
- **럭스:** 1제곱미터의 면적에 1루멘을 방출하는 광원의 조도가 1럭스입니다. 실제로 광원을 측정할 때 보통 럭스를 사용하며, Unity에서는 방향 광원과 함께 자주 사용하게 됩니다.
- **니트:** 제곱미터당 1칸델라에 해당하는 휘도 단위입니다. 디스플레이 기기와 LED 패널(텔레비전 또는 모니터 등)은 밝기를 측정할 때 주로 니트 단위를 사용합니다.

- **EV100:** 100 ISO 필름의 노출 값인 EV100에 해당하는 강도를 의미합니다. 위의 노출 값 수식을 참조하세요. 노출을 늘리면 로그의 특성으로 인해 조명이 두 배가 됩니다.
- 실제 광원을 재현하려면 기술 사양에 나와 있는 단위로 바꾸고 올바른 광속이나 휘도를 입력하세요. HDRP는 입력된 값을 PLU에 맞게 변환하므로 강도를 설정할 때 지레짐작으로 설정하지 않아도 됩니다.
- 아이콘을 클릭하여 **Exterior, Interior, Decorative, Candle** 등의 프리셋을 선택할 수도 있습니다. 구체적인 값을 설정하지 않는 경우 프리셋 설정을 기준으로 활용하기 좋습니다.

일반적인 조명 및 노출 값

다음 요약 자료에는 일반적으로 사용하는 실제 **광원**의 색 온도 값과 빛 강도 값이 포함되어 있으며, 여러 조명 시나리오에 대한 **노출** 값도 포함되어 있습니다.

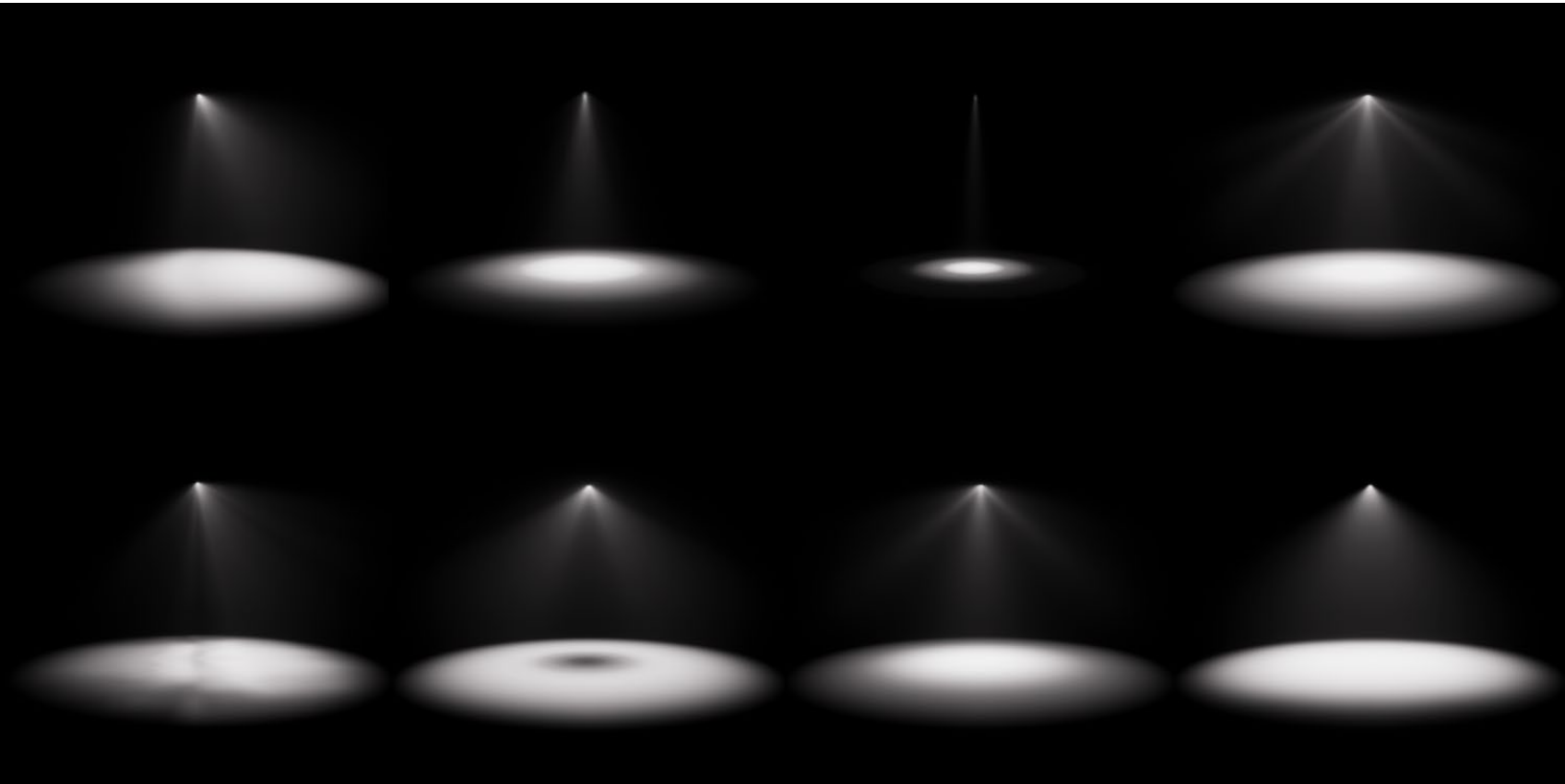


조명 및 노출 수준 가이드

물리 광원 단위 [기술 자료](#)에서 일반적인 조명 값에 대한 전체 표를 확인할 수 있습니다.

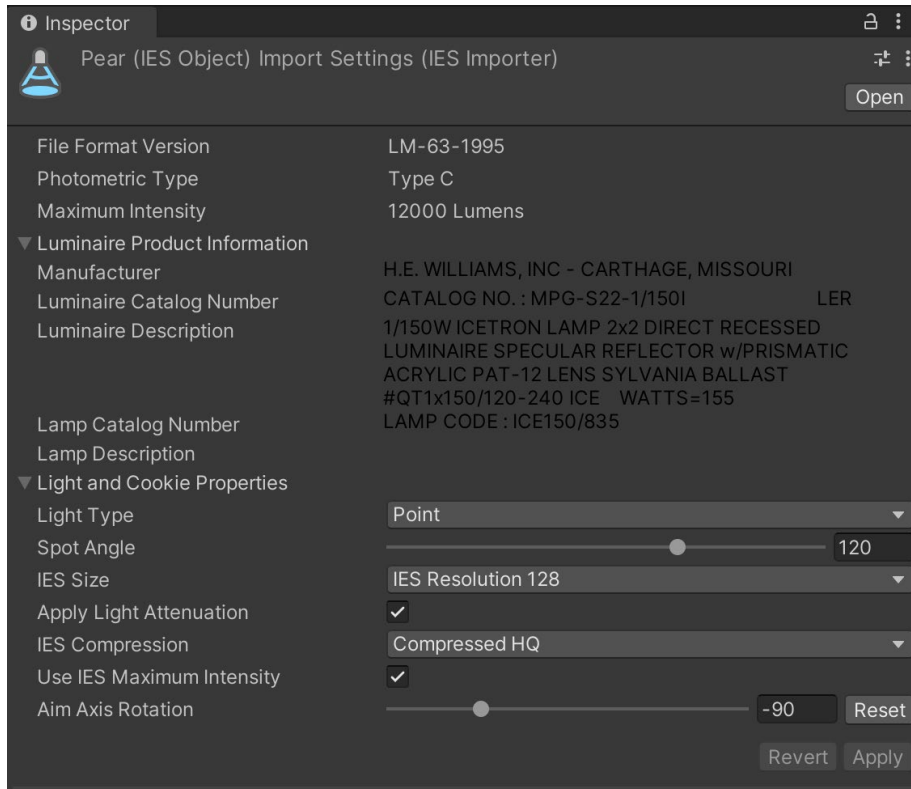
IES 프로파일 및 쿠키

IES 프로파일을 사용하면 점 광원, 스포트 광원, 면 광원을 실제 광원의 감쇠와 더 비슷하게 표현할 수 있습니다. 이는 [광원 쿠키](#)처럼 작동하여 광원 패턴에 특정 제조업체의 사양을 적용합니다. IES 프로파일을 사용하면 조명에 사실적인 느낌을 더할 수 있습니다.



다양한 조명에 적용된 IES 프로파일

IES 프로파일은 **Assets > Import New Asset**에서 임포트합니다. 그러면 [임포터](#)에서 적절한 강도가 설정된 [Light](#) 프리팹을 자동으로 생성합니다. 생성된 프리팹을 씬 뷰나 계층 구조 창으로 드래그하고 색 온도만 조정하면 됩니다.



IES 프로파일 및 임포트 설정

IES 프로파일을 알아볼 수 있는 몇 가지 좋은 출처는 다음과 같습니다.

실제 제조업체

- [Philips](#)
- [Lithonia Lighting](#)
- [Efficient Lighting Systems](#)
- [Atlas](#)
- [ERCO](#)
- [Lamp](#)
- [Osram](#)

아티스트 소스

- [Renderman](#)

IES 프로파일 임포터에 대한 자세한 내용은 [기술 자료](#)를 참조하시기 바랍니다.

HDRP 전역 조명

전역 조명의 이해

현실 세계에서는 빛이 표면에 닿으면 멈추지 않고 반사, 굴절, 산란 등이 일어납니다. 현실의 공간에서 광자가 흩어지지 못하도록 붙잡을 수 있는 것은 블랙홀뿐입니다. 따라서 콘크리트, 모래, 돌과 같이 윤기가 없는 소재와 닿아도 빛은 반사됩니다.

사람의 눈은 이러한 빛의 미묘한 특성에 익숙해져 있습니다. 빛이 여러 복잡한 머티리얼과 상호 작용하는 방식이 정확하게 표현되었을 때 우리는 렌더링이 사실적이라고 느낍니다.

GI(전역 조명)의 목적은 이것을 모방하는 것입니다. GI가 없다면 직접광을 받지 못하는 영역은 기본적으로 어둡게 표현됩니다. GI는 빛의 난반사를 추정하여 현실 세계에서 빛의 색상이 한 표면에서 다른 표면으로 전파되는 현상을 모방합니다. 이렇게 반사된 간접 조명은 게임에 현실감을 더해 줍니다.

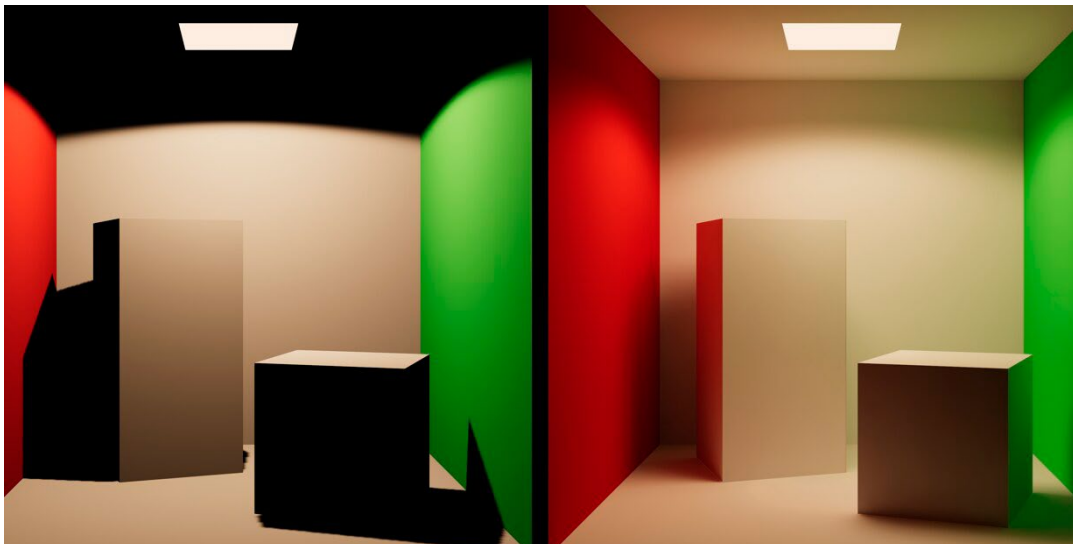


전역 조명으로 사실적인 결과를 얻을 수 있습니다. 출처: ArchVizPro Vol. 10

HDRP 전역 조명 기능

GI는 하나의 기술이 아니라 이러한 현상을 표현하는 데 도움이 되는 알고리즘의 모음입니다. Unity는 물리적 환경에서 빛의 운동을 재현하기 위한 톨의 전체 생태계를 제공합니다. 사실적인 렌더링을 위해 조합해서 사용하게 되는 기술은 주로 다음과 같습니다.

- **베이킹된 전역 조명(베이킹된 라이트맵):** 정적 오브젝트에 대한 간접광을 사전에 계산하고 런타임에 사용하기 위해 결과를 저장합니다.
- **실시간 전역 조명:** 씬 변화에 대응하여 실시간으로 간접광을 계산합니다.
 - SSGI(스크린 공간 전역 조명): 실시간 빛 전파에 사용
 - RTGI(레이트레이싱 전역 조명): 물리적으로 정확한 빛 반사에 사용



오른쪽의 전역 조명은 난반사나 간접 조명을 시뮬레이션합니다.



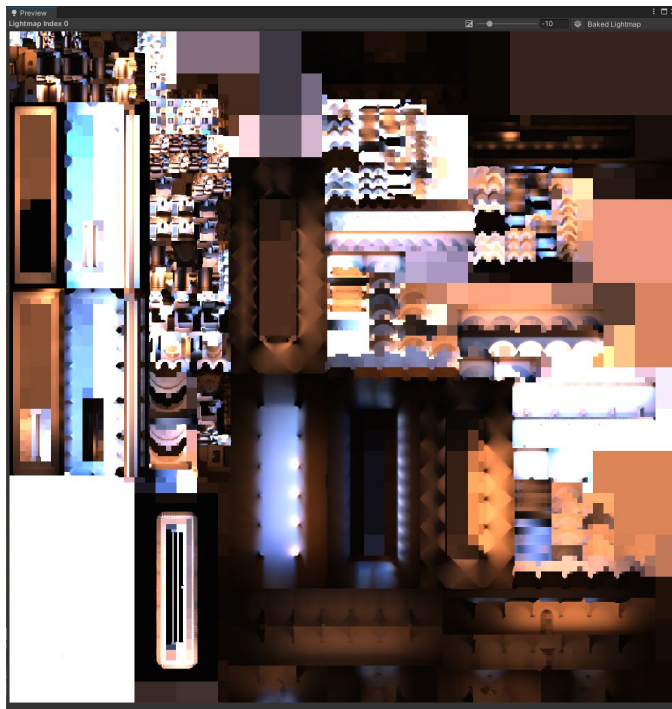
- **라이트 프로브와 APV(적응적 프로브 볼륨):** 동적 오브젝트가 실시간 계산 없이 간접 조명을 받을 수 있도록 미리 계산된 조명 정보를 저장합니다.
 - 라이트 프로브: 베이킹된 광원을 캡처하여 수동으로 배치된 지점에 보관
 - APV: 동적으로 프로브 밀도를 조정하고 더욱 정확한 대규모 조명을 위해 오클루전 지원
- **환경 조명:** 이미지 기반 기법을 사용하여 HDR(High Dynamic Range) 이미지(예: HDRI 스카이박스)로 환경 조명을 시뮬레이션함으로써 사실적인 주변광을 제공합니다.
- **실시간 레이트레이싱:** 빛과 머티리얼 간의 고도로 사실적인 상호 작용을 구현하여 세부적인 반사와 그림자를 포착합니다.
- **패스트레이싱(Path tracing):** HDRP가 통합된 프로세스에서 다양한 효과(그림자, 반사, 굴절, 간접 조명 등)를 계산할 수 있도록 카메라로부터 광선을 재현합니다.

HDRP의 전역 조명은 정적 환경과 동적 환경을 모두 지원합니다. 이를 통해 게임 월드가 더욱 생동감 있게 반응합니다.

더 자세히 알아보려면 ‘[Unity에서 조명 환경을 구현하는 4가지 기법](#)’ 동영상(영문)을 시청해 보세요.

베이킹된 전역 조명

베이킹된 GI(전역 조명)는 씬 내 광원의 상호 작용을 사전에 계산하여 결과를 라이트맵이라는 텍스처로 저장합니다. 이를 통해 런타임에 계산하지 않고도 빛이 표면에서 어떻게 반사되고 굴절되는지 캡처할 수 있습니다.

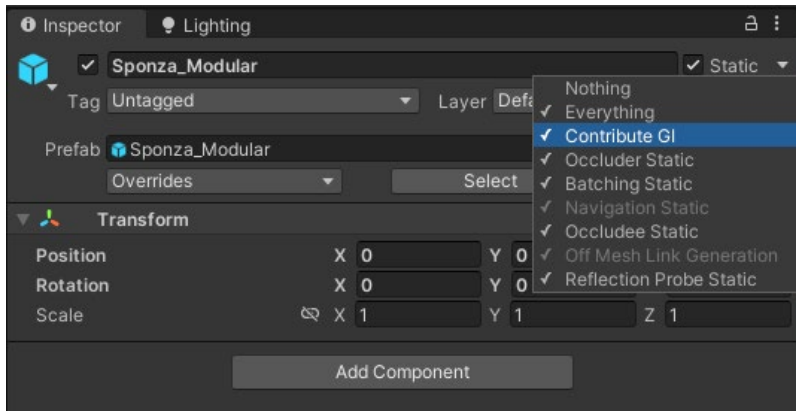


베이킹된 라이트맵

모바일 플랫폼용으로 개발할 때 게임 환경에 사실적인 조명을 추가하기 위해 흔히 사용되는 방법입니다. 조명을 베이킹할 때 리소스를 많이 소모하는 계산은 오프라인에서 한 번만 수행됩니다.

따라서 런타임에 이러한 조명을 계산하기 위해 추가로 성능을 소모하지 않습니다.

베이킹된 GI에는 다양한 장점이 있는데, 우선 광원을 사전에 계산하여 런타임 시 필요한 작업량을 크게 줄임으로써 성능을 향상합니다. 이 프로세스를 통해 소프트 색도우나 미묘한 난반사와 같은 복잡한 뉘앙스를 캡처할 수 있습니다. 또한 실시간 조명 계산에서 흔히 발생하는 많은 시각적 결함을 방지합니다.



라이트매핑은 정적 오브젝트에만 사용할 수 있습니다.

하지만 사전 베이킹된 조명은 정적 요소의 전역 조명만 캡처할 수 있습니다. 씬의 동적이거나 움직이는 요소에는 사전 계산된 조명 데이터의 이점을 충분히 활용할 수 없습니다.

라이트맵은 텍스처 에셋과 마찬가지로 추가 메모리 및 디스크 공간이 필요합니다. 모바일 환경 등 메모리 리소스가 제한된 플랫폼에서 작업하는 경우 이 요구 사항에 주의하세요.

마지막으로, GI를 베이킹하는 작업은 씬의 복잡도와 사용하는 하드웨어에 따라 오랜 시간이 소요될 수 있습니다. 정적 지오메트리가 많은 대규모 환경의 라이트매핑은 몇 분에서 몇 시간까지 걸릴 수도 있습니다.

라이트매핑 워크플로

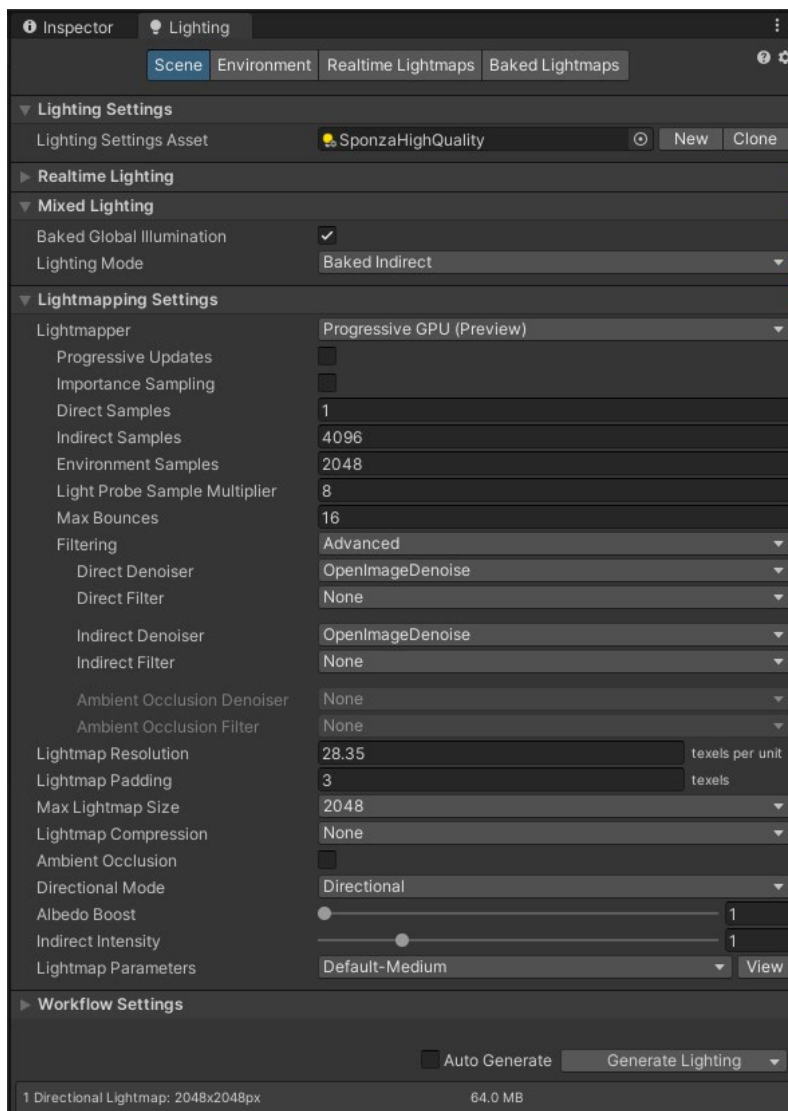
베이킹된 GI로 씬을 라이트매핑하는 방법은 다음과 같습니다.

1. **광원 모드 설정:** 베이킹 프로세스에 포함할 광원은 모두 **Baked** 또는 **Mixed** 모드로 설정해야 합니다.
2. **오브젝트를 정적 라이트맵으로 표시:** 또한 베이킹 프로세스에 포함할 오브젝트는 **Static** 드롭다운 메뉴에서 **Contribute GI** 또는 **Static Everything**로 플래그를 지정해야 합니다.
3. **새 조명 설정 에셋 생성:** Lighting 창에서 **LightingSettings** 클래스의 저장된 인스턴스를 나타내는 새 조명 설정 에셋을 생성합니다. 베이킹된 GI(및 아래 설명할 Enlighten Realtime GI)의 사전 계산된 조명 데이터가 이 에셋에 저장됩니다.

4. 조명 모드 선택: **Lighting Settings Asset**에서 **Bake Indirect**, **Shadowmask** 또는 **Subtractive**를 선택합니다. 이 가이드를 참조하여 **조명 모드**를 결정하세요.
5. **Lightmap Settings**에서 기본 파라미터 조정: **Lightmap Resolution**과 **Sample Count**로 베이킹 정확도와 품질을 조정합니다. 일반적으로 해상도가 높고 샘플 수가 많을수록 더 뛰어난 품질의 결과를 얻을 수 있지만 베이킹 시간과 리소스 소모량이 늘어납니다.
6. **Generate Lighting**을 클릭하여 라이트매핑을 시작합니다.

베이킹된 라이트매핑은 계산하면서 라이트맵을 점진적으로 다듬는 **프로그레시브 라이트매퍼**를 사용합니다.

베이킹 프로세스가 진행되는 동안 결과를 미리 볼 수 있습니다. 필요하다면 베이킹을 중단하고 설정을 조정한 다음 다시 베이킹합니다. 이러한 반복 작업(iteration)을 통해 더욱 인터랙티브한 조명 워크플로를 구축할 수 있습니다. 결과는 **Lighting** 창의 **Baked Lightmaps** 탭에서 확인할 수 있습니다.



프로그레시브 라이트매퍼를 활성화합니다.



라이트맵 최적화

베이킹된 GI 최적화는 시각적 품질과 컴퓨팅 효율성, 메모리 관리의 균형을 전략적으로 맞추는 작업입니다.

라이트매핑 시 참고할 만한 일반적인 팁은 다음과 같습니다.

- **라이트맵 해상도:** 해상도가 높을수록 더 많은 디테일을 캡처할 수 있지만 메모리 사용량이 늘어납니다. 우선순위를 정하여 중요한 오브젝트는 해상도를 높이고, 배경 요소는 해상도를 낮추는 게 좋습니다.
- **텍셀 낭비 방지:** 자갈이나 전선같이 작거나 얇은 오브젝트는 라이트맵 리소스의 균형을 망가뜨릴 수 있습니다. 씬 조명에 큰 영향을 미치는 오브젝트(예: 색상이 밝거나 발광 머티리얼이 포함된 오브젝트)가 아니라면 Static 메뉴 또는 MeshRenderer에서 **Contribute Global Illumination**을 비활성화하여 GI 계산에서 제외합니다.



작거나 얇은 오브젝트에 텍셀을 낭비하지 않는 것이 좋습니다.

- **샘플링:** 샘플 수는 조명 베이킹의 품질에 직접적인 영향을 미칩니다. 샘플 수가 많을수록 조명 디테일이 풍부해지지만 베이킹 시간이 늘어납니다.
- **노이즈 제거:** 조명이 어두운 등의 특정 상황에서는 베이킹 시 비주얼 노이즈가 발생할 수 있습니다. **Auto**를 선택하면 HDRP가 자동으로 노이즈 제거 알고리즘을 선택합니다. 또는 **Advanced**를 선택하고 Direct Denoiser와 Indirect Denoiser를 선택할 수 있습니다.
- **라이트맵 압축:** 압축 기법은 메모리 사용량을 줄이지만 품질이 약간 저하될 수 있습니다.
- **안티앨리어싱:** 성능을 최적화하려면 안티앨리어싱 수준을 낮추는 것을 고려해 보세요. 예를 들어, **Project Settings > Quality**에서 8x Multi Sampling을 2x Multi Sampling으로 낮출 수 있습니다.



GPU 라이트매핑

프로그레시브 라이트매퍼는 두 가지 백엔드 옵션이 있는데, 바로 CPU와 GPU입니다. [프로그레시브 GPU 라이트매퍼](#)는 컴퓨터의 GPU와 전용 VRAM(Video Ram)을 사용하여 베이킹된 라이트맵 생성 시간을 단축합니다.

사용 가능한 경우, GPU 라이트매퍼는 CPU 라이트매퍼에 비해 조명 데이터를 10배 이상 빠르게 생성합니다. Unity 6의 GPU 라이트매퍼는 정식 제작에 사용 가능하며, 다음과 같은 부분이 개선되어 있습니다.

- **LightBaker v1.0:** 이 새로운 백엔드는 베이킹 시 씬 상태를 즉각적인 스냅샷으로 캡처하여 더욱 예측 가능하고 안정적인 프로세스를 지원합니다. 베이킹 중에 씬을 변경할 경우, 더 이상 베이킹 프로세스가 재시작되지 않고 에디터가 한층 더 높은 반응성을 유지합니다.
- **GPU 베이킹 프로파일:** GPU 베이킹 프로파일을 사용하면 성능과 GPU 메모리 사용량 사이의 균형을 이룰 수 있습니다.
- **더 적은 GPU 메모리 요구 사항:** Unity 6의 새로운 베이킹 백엔드는 최소 GPU 메모리 요구 사항을 2GB VRAM으로 줄여 줍니다.

LightBaker 백엔드가 라이트맵, 라이트 프로브, 새도우 마스크, 앰비언트 오클루전 텍스처의 처리를 개선합니다. 이 새로운 아키텍처는 내부 구조가 더 모듈화되어 있어 유지 관리와 디버깅이 쉽습니다. 이 개선 사항은 결과의 품질을 유지하면서 조명 반복 작업(iteration) 시간을 단축하도록 설계되었습니다.

Auto 모드는 기존 프로그레시브 베이킹 기능을 사용합니다.

GPU 라이트매퍼를 사용할 때, 베이킹 속도를 최적화할 수 있도록 참고할 만한 내용은 다음과 같습니다.

- 다른 GPU 가속 애플리케이션, 특히 VRAM을 사용하는 애플리케이션을 종료합니다.
- Intel Open Image와 같은 CPU 기반 디노이저로 전환하여 VRAM을 확보합니다.
- GPU가 여러 개인 경우, 렌더링과 베이킹에 각각 GPU를 하나씩 할당합니다.
- 특히 라이트맵 크기가 4096 이상인 경우, 안티앨리어싱 샘플 수를 줄입니다.

매뉴얼에서 [성능 가이드라인](#)을 참고하세요.

Lightmap UVs

라이트맵은 텍스처이므로 Unity가 씬에서 라이트맵을 올바르게 사용하려면 UV가 필요합니다.

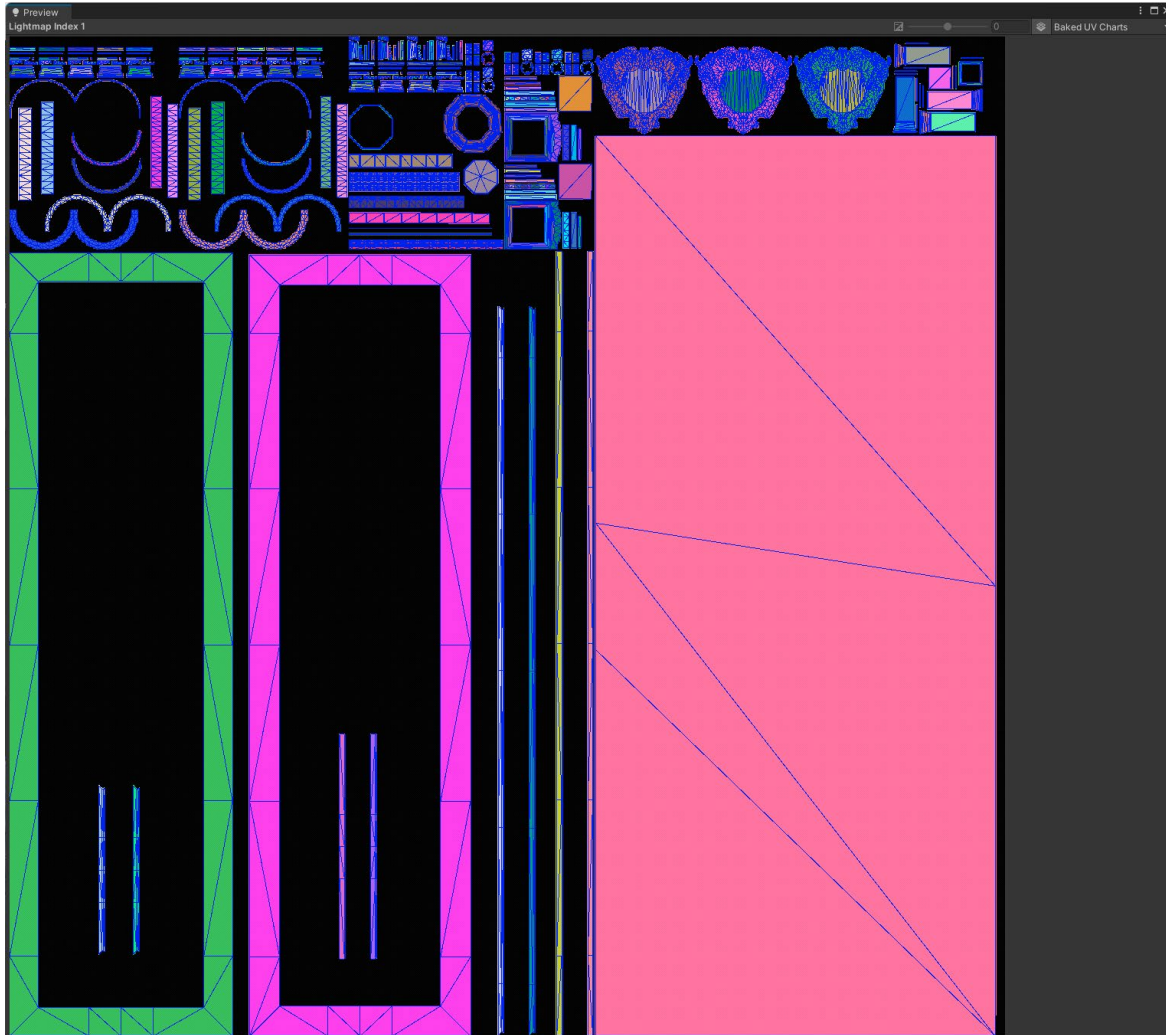
Unity는 베이킹된 전역 조명과 실시간 전역 조명에 대해 별도의 라이트맵 UV 세트를 사용합니다. 이는 인스턴스 그룹화와 메시 스케일링 측면에서의 차이 때문입니다. UV는 메시별로 적용되므로 동일한 메시를 사용하는 모든 인스턴스는 같은 UV를 공유합니다.



Unity는 모델을 임포트할 때 UV를 생성하거나 모델 내 기존 UV 세트를 사용할 수 있습니다. Unity는 베이킹된 라이트맵 UV를 Mesh.uv2 채널에 저장합니다. 이 채널은 TEXCOORD1 셰이더 시맨틱에 매핑되며 보통 'UV1'이라고 부릅니다.

Unity는 번짐 효과(한 UV 영역의 빛이 다른 영역으로 번지는 효과)와 같은 그래픽 문제를 줄이기 위해 각 차트의 경계에 약간의 패딩(padding)이 존재하도록 실시간 라이트맵 UV를 리패킹합니다.

UV 계산은 인스턴스의 스케일과 라이트맵 해상도에 의존합니다. Unity는 가능한 경우 메시, 스케일, 라이트맵 해상도가 동일한 Mesh Renderer 컴포넌트들끼리 UV를 공유하도록 최적화합니다.



라이트맵은 UV가 있어야 올바르게 표시할 수 있습니다.

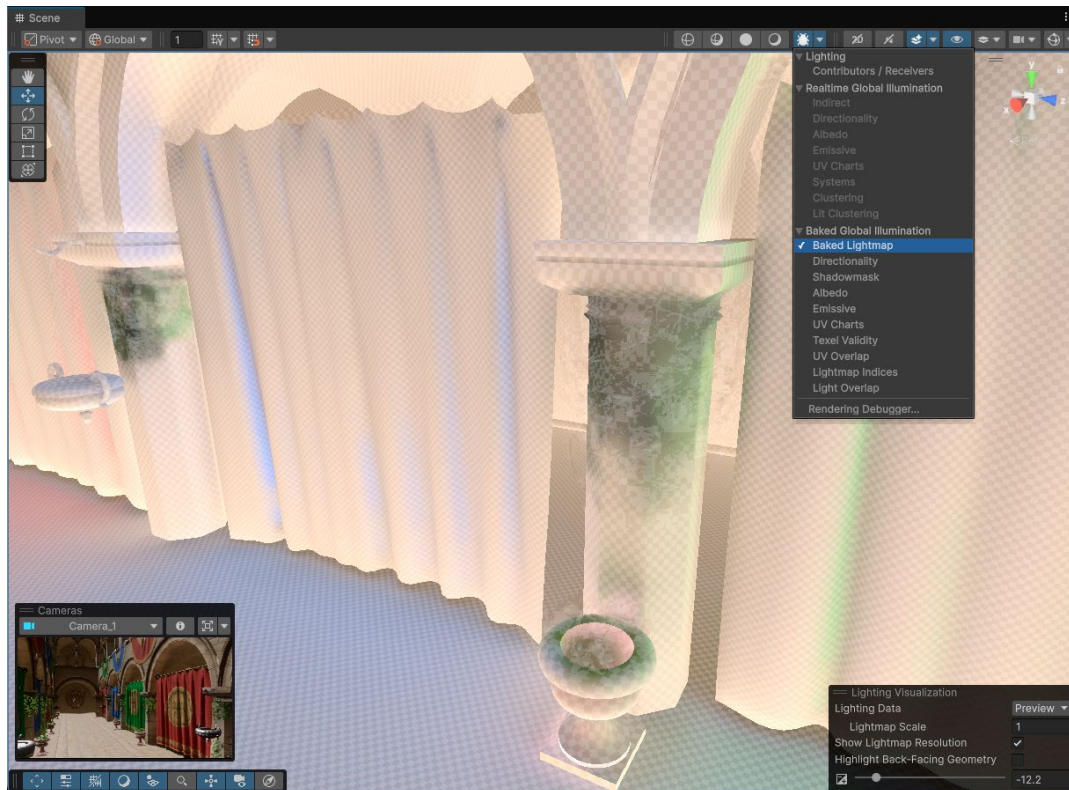
라이트맵 UV에 대한 자세한 기술 정보는 [기술 자료 페이지](#)를 참조하세요.



조명 베이킹 인터랙티브 프리뷰(Unity 6.1)

Unity 6.1에서는 씬 뷰에 인터랙티브 **GI 디버그 프리뷰 모드**가 도입되었습니다. 이 새로운 기능을 사용하면 베이킹된 조명의 실시간 프리뷰를 통해 씬의 변화가 전역 조명에 어떤 영향을 주는지를 확인할 수 있으며, 이때 전체를 다시 베이킹할 필요가 없습니다.

씬 뷰 드로우 모드에 직접 통합되어 있는 이 기능은 기존의 베이킹된 데이터를 변경하지 않으면서 업데이트를 인터랙티브 방식으로 표시합니다. 이 기능을 사용하려면 드롭다운에서 **Baked Global Illumination** 아래에 있는 디버그 드로우 모드를 선택하세요.



씬 뷰에서 GI 디버그 프리뷰 모드를 사용할 수 있습니다.

이 기능은 기존의 베이킹된 GI(전역 조명)에 대한 **Auto Generate** 시스템을 대체합니다.

다이내믹 전역 조명

베이킹된 GI는 고품질 간접 조명을 제공하긴 하나, 설정에 시간이 소요되고 컴퓨트 비용이 높을 수 있습니다. 게다가 정적이라는 중대한 단점이 있습니다. 베이킹된 라이트맵은 변화하는 광원 조건이나 움직이는 오브젝트, 또는 하루 동안의 시간 변화에 대응할 수 없습니다. 이 문제를 해결하기 위해, Unity 6 HDRP에는 실시간 또는 실시간에 가까운 조명 업데이트를 제공하는 여러 다이내믹 GI(전역 조명) 기법이 있습니다.

SSGI(스크린 공간 전역 조명)는 화면의 덤스 및 색상 버퍼를 사용해 반사된 산란광을 계산합니다. 따라서 미리 계산된 라이트맵이 없어도 실시간 간접 조명을 사용할 수 있습니다.

RTGI(레이트레이싱 전역 조명)은 보다 물리적으로 정확한 광원 반사를 위해 하드웨어 가속 레이트레이싱 (DXR)을 사용하여 SSGI를 확장합니다. 레이트레이싱이 활성화되면 RTGI 설정이 반영되도록 인스펙터의 사용 가능한 프로퍼티가 바뀝니다.

라이트 프로브와 APV(적응적 프로브 볼륨)은 프로브 기반 간접 조명을 사용하여 전역 조명을 모방합니다. APV는 프로브를 적응식으로 배치하여 실시간 또는 실시간에 가까운 간접 조명 업데이트를 제공하기 때문에 조명이 동적으로 변화하는 대규모 환경에 적합합니다. 라이트 프로브와 APV에 대한 자세한 내용은 [아래 섹션](#)을 참조하세요.

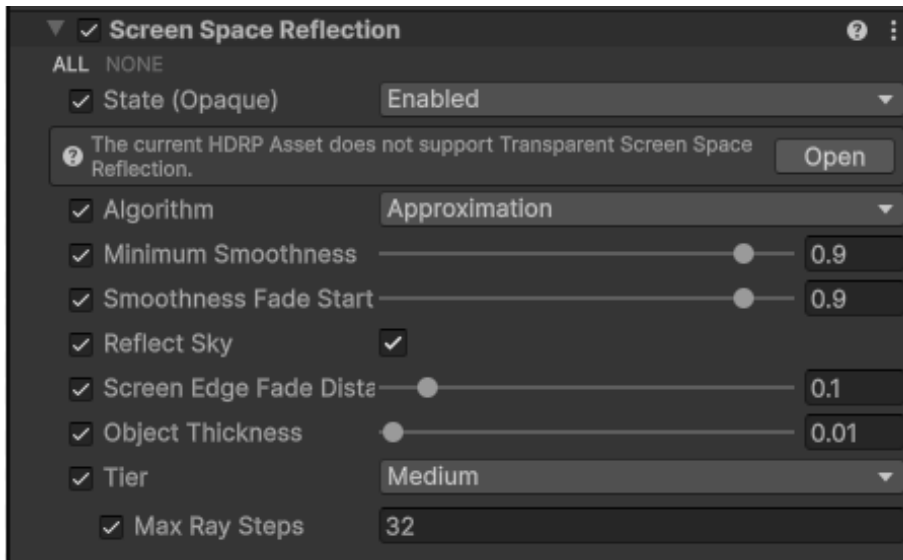
SSGI와 RTGI는 모든 **라이트맵** 및 **라이트 프로브** 데이터를 대체합니다. 카메라에 영향을 주는 볼륨에서 활성화된 경우, 라이트 프로브와 앰비언트 프로브는 더 이상 게임 오브젝트의 간접 조명에 기여하지 않습니다.

프로젝트의 성능 요구 사항과 다이내믹 조명 요구 사항에 가장 적합한 기법을 선택하세요. 위에서 소개한 기법은 프로젝트의 성능 요구 사항과 다이내믹 조명 요구 사항에 따라 베이크된 광원을 보완하거나 대체할 수 있습니다.

SSGI(스크린 공간 전역 조명)

SSGI는 뎀스 및 색상 버퍼로 빛이 표면에서 반사되는 방식을 계산하여 간접 조명을 모방합니다. 빠르게 변화하는 광원이 있고 라이트맵을 사전에 계산할 수 없는 경우에 유용합니다.

SSGI는 화면 내 데이터에만 작동하며, 화면 밖의 오브젝트는 전역 조명에 기여하지 않습니다.



스크린 공간 전역 조명을 볼륨 오버라이드로 추가합니다.

볼륨 시스템에서 **Screen-Space Global Illumination** 오버라이드를 추가합니다(**Lighting > Screen Space Global Illumination**). **Frame Settings**와 HDRP 에셋의 **Lighting** 섹션에서 이 기능을 활성화합니다.



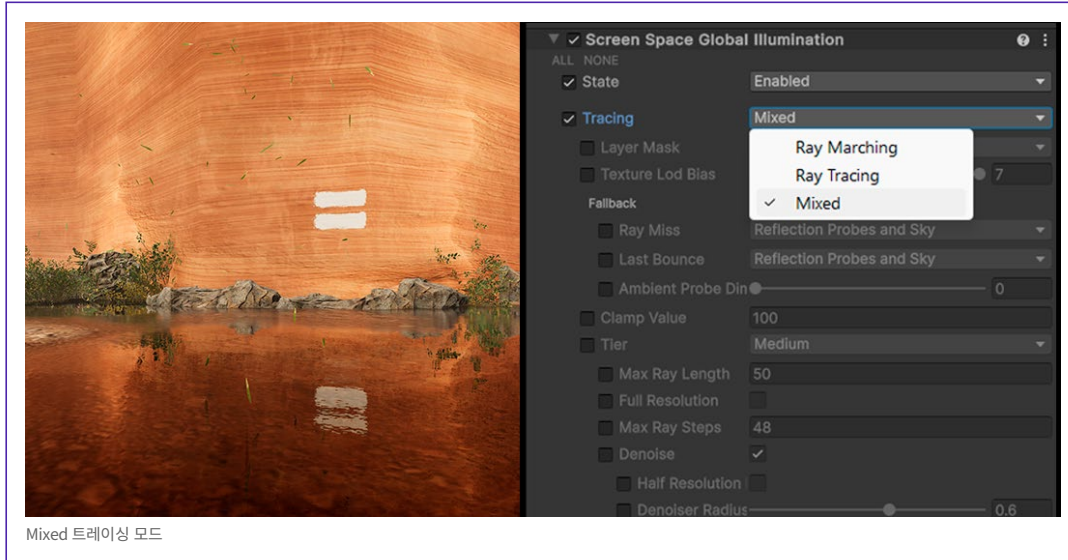
스크린 공간 전역 조명은 사실적인 간접 조명을 투사합니다.

트레이싱 모드

SSGI와 스크린 공간 반사는 둘 다 **트레이싱** 모드를 사용합니다. 트레이싱 모드에 따라 인스펙터에 나타나는 프로퍼티가 달라집니다.

- **Ray Marching:** 이 모드는 스크린 공간 **레이 마칭** 솔루션을 사용합니다. 레이 마칭은 스크린 공간에 표시되며 GBuffer에 나타나는 오브젝트의 조명을 결정합니다.
- **Ray Tracing:** 이 모드는 레이트레이싱을 사용합니다. 레이트레이싱은 화면 외부의 오브젝트 또는 레이 마칭 한도를 넘어서는 오브젝트의 조명을 계산합니다. 자세한 내용은 아래의 레이트레이싱과 패스트레이싱을 참조하세요.
- **Mixed 트레이싱 모드:** 레이트레이싱과 레이 마칭이 둘 다 사용됩니다. 이 모드에서는 파티클, 버텍스 애니메이션, 데칼 등 레이트레이싱만으로는 캡처할 수 없는 추가적인 효과를 캡처합니다. Mixed 트레이싱 모드를 사용하려면 **Performance** 모드와 **Lit Shader Mode**를 **Deferred**로 설정해야 합니다. 이 모드는 디퍼드 렌더링으로 렌더링된 GBuffer에 있는 오브젝트에만 영향을 줍니다.

Unity 6에서는 투명 스크린 공간 반사의 Mixed 트레이싱 모드가 레이트레이싱된 반사만 사용하는 대신 두 가지 트레이싱 모드를 함께 사용합니다. 예를 들어, **이 씬**에는 앞사귀 파티클, 흰색 데칼, 절벽 면의 변형되지 않은 지오메트리에서는 보이지 않는 게임 오브젝트에 대한 반사도 포함될 수 있습니다.

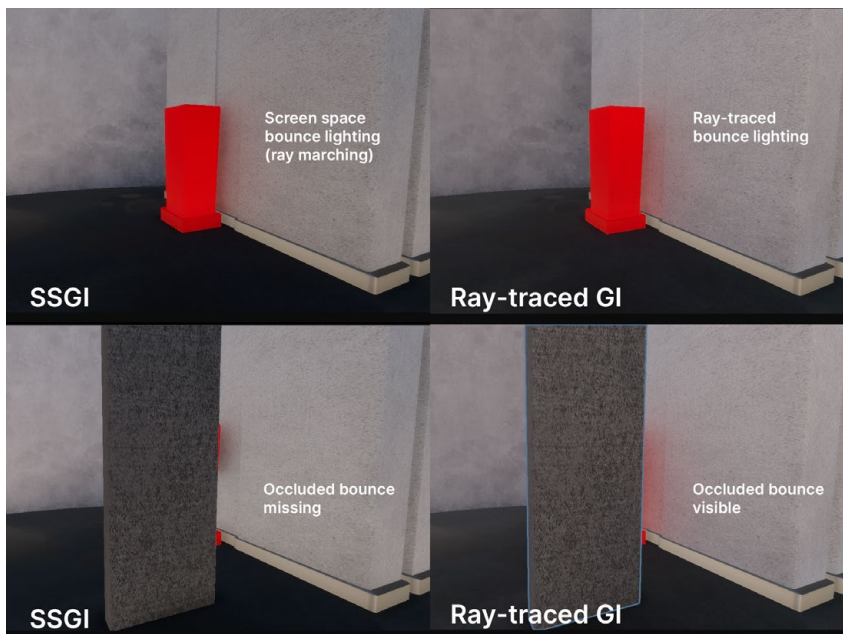


RTGI(레이트레이싱 전역 조명)

RTGI는 하드웨어 가속 레이트레이싱을 사용하여 간접광을 계산합니다. 이는 SSGI보다 높은 정확도를 제공하며 화면 밖의 오브젝트에도 적용되지만 컴퓨팅 비용이 높습니다.

RTX 케이블 GPU가 있는 플랫폼의 하이엔드 실시간 렌더링에는 레이트레이싱이 가장 적합합니다.

HDRP 설정에서 **Ray-traced global illumination**을 활성화하고, HDRP 에셋에서 DXR 지원을 활성화하세요.



레이트레이싱 전역 조명은 화면 밖의 오브젝트에도 적용됩니다.

자세한 내용은 [레이트레이싱 챕터](#)를 참조하세요.



Enlighten Realtime GI 지원 중단

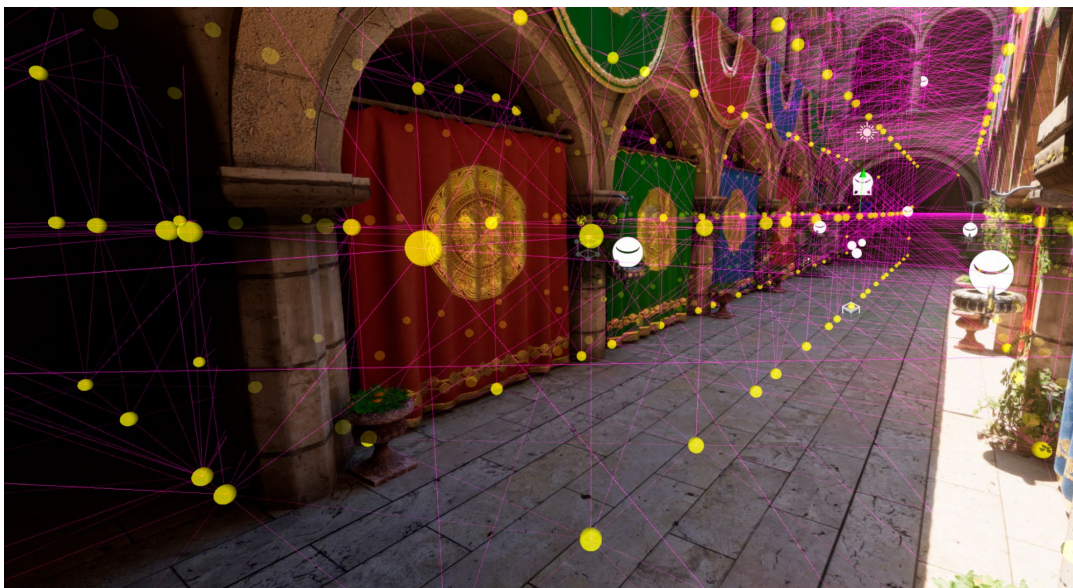
Unity 6는 (Lighting > Realtime Global Illumination 설정의) **Enlighten Realtime GI**가 마지막으로 지원되는 릴리스입니다. 자세한 내용은 [2021년 전역 조명 업데이트](#) 포럼 게시물을 참조하세요.

라이트 프로브와 적응적 프로브 볼륨

베이킹된 전역 조명과 실시간 전역 조명으로 정적 환경에 사실적인 조명을 추가할 수 있습니다. 반면 움직이는 오브젝트에는 라이트 프로브라는 다른 조명 기술을 사용할 수 있습니다.

라이트맵과 유사하게 라이트 프로브는 씬의 조명에 대한 ‘베이킹된’ 정보를 저장합니다. 라이트맵은 씬의 표면에 닿는 광원에 대한 정보를 저장하지만, 라이트 프로브는 빈 공간을 지나가는 광원에 대한 정보를 저장한다는 점에서 차이가 있습니다.

라이트 프로브를 사용하여 디테일 수준(LOD) 시스템으로 움직이는 오브젝트나 정적 배경에 빛을 비춰 보세요. 라이트 프로브는 간접 반사광을 비롯한 고품질 광원을 캡처합니다.



라이트 프로브는 빈 공간에서 조명을 샘플링합니다.

라이트 프로브와 SSGI/RTGI 비교

카메라에 영향을 주는 볼륨 오버라이드에서 스크린 공간 전역 조명 또는 레이트레이싱 전역 조명을 활성화하면 **라이트 프로브**와 앰비언트 프로브가 더 이상 해당 볼륨 내에 있는 게임 오브젝트의 조명에 기여하지 않습니다(자세한 내용은 ‘다이나믹 전역 조명’ 참조).

단, APV(적응적 프로브 볼륨)는 스크린 공간 효과가 적용되지 않는 영역에 간접 조명 정보를 제공하여 SSGI 및 RTGI를 보완할 수 있습니다.

라이트 프로브 그룹

Light Probe Group 컴포넌트를 사용하여 기본적인 광원 프로브 설정에 대해 알아볼 수 있습니다.

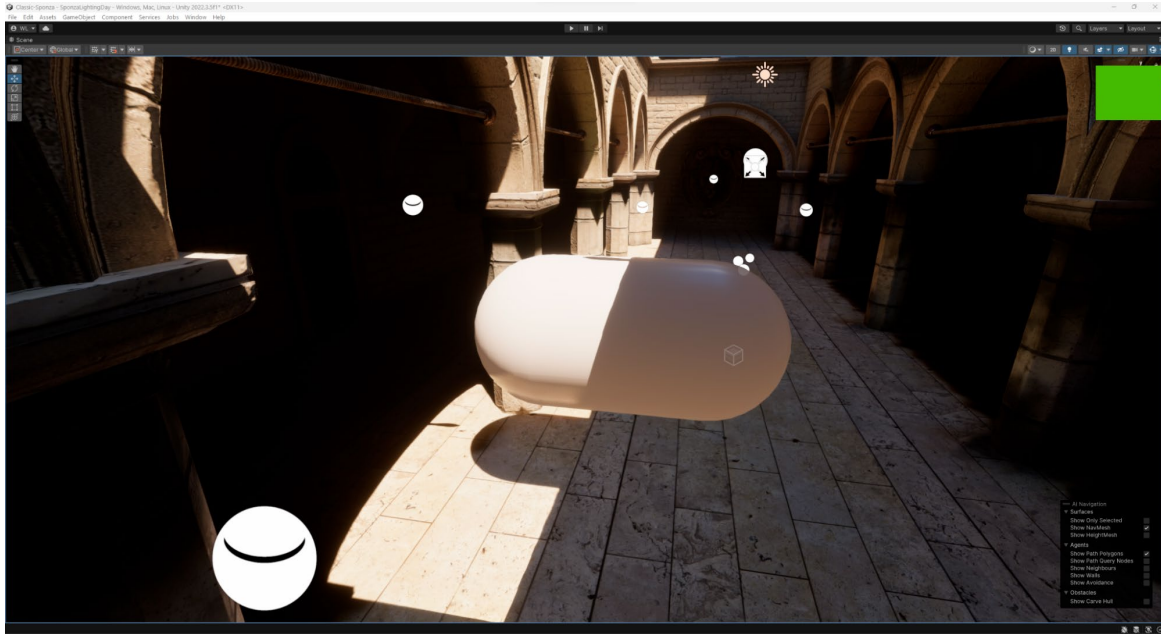
씬에 라이트 프로브를 배치하려면 빈 게임 오브젝트에 Light Probe Group 컴포넌트를 추가(**Component > Rendering > Light Probe Group**)합니다. 그런 다음 컴포넌트 설정에서 프로브의 위치를 지정하고, 프로브를 추가하거나 제거할 수 있습니다.

동적 오브젝트가 이동할 수 있는 영역에서 조명 변화가 뚜렷한 곳(예: 출입구 근처, 실내와 실외 영역 사이)에 라이트 프로브를 그리드나 클러스터로 배치합니다.

라이트 프로브를 배치하고 나면 베이킹해야 씬의 주변광을 캡처할 수 있습니다. 베이킹된 라이트맵과 마찬가지로 Lighting 창의 **Generate Lighting** 버튼을 누릅니다.

프로브를 사용한 조명은 런타임 시 상대적으로 리소스를 적게 소모하고 사전 계산이 빠르게 완료됩니다. 라이트 프로브는 언래핑하는 데 오래 걸리고 메모리 사용량이 많은 라이트맵을 필요로 하지 않습니다.

하지만 라이브 프로브로 조명을 적용한 오브젝트는 가장 가까운 프로브 4개의 결과를 블렌딩하므로 가끔 조명이 부정확하게 표현될 수 있습니다. 아래 이미지에서는 라이트 프로브를 통과하여 움직이는 캡슐이 씬의 밝은 부분과 어두운 부분 사이를 지날 때 부자연스러운 조명이 나타납니다.



라이트 프로브는 리소스 소모량이 적지만 부정확한 결과를 낼 수도 있습니다.

라이트 프로브는 계산이 빠르게 완료되지만 직접 배치해야 합니다. 조명 환경 샘플링은 반복적인 작업이며 많은 양의 작업이 요구될 수 있습니다. 그러니 씬에서 라이트 프로브를 하나씩 정렬하는 대신 적응적 프로브 볼륨(아래 참조)을 사용하는 방안을 고려해 보세요.

라이트 프로브 그룹에 대한 자세한 내용은 [라이트 프로브 기술 자료 페이지](#)를 참조하세요.

런타임에 라이트 프로브 이동

Unity 6에는 정식 제작을 위해 라이트 프로브 작업을 간소화하는 새로운 기능이 있습니다.

새로운 [LightProbes API](#)를 사용하면 스크립트를 사용하여 런타임에 기존의 라이트 프로브를 재배포할 수 있습니다. 애플리케이션이 모듈식 환경 또는 절차적으로 생성된 환경을 사용하는 경우에 유용합니다.

예를 들어, 게임 월드가 여러 개의 씬으로 분할되어 있고 런타임에 씬이 추가로 로드된다면, 환경의 여러 부분을 제자리로 이동해야 할 수 있습니다. 이때 각 씬에 해당하는 라이트 프로브도 재배포해야 합니다. 그렇지 않으면 조명이 일치하지 않게 됩니다.

그러나 런타임에 라이트 프로브 그룹의 트랜스폼을 조정하는 것만으로 라이트 프로브를 재배포할 수는 없습니다. 트랜스폼 조정은 베이킹 위치에만 영향을 주기 때문입니다. 새로운 LightProbes API를 사용하면 프로브 데이터를 클로닝하고 프로브 위치를 업데이트한 다음 스크립트를 통해 프로브 네트워크를 다시 구축할 수 있습니다. API에는 복수의 프로브를 한 번에 효율적으로 이동하기 위한 여러 성능 최적화도 포함되어 있습니다.

따라서 작은 씬으로부터 환경을 만드는 경우, 조명을 사전에 베이킹한 다음 런타임에 조명의 일관성을 유지하면서 게임 월드의 모듈식 부분들을 재배포할 수 있습니다.

구현 예시는 [Unity 라이트 프로브 런타임 이동 기술 자료](#)를 참조하세요.

적응적 프로브 볼륨

Unity 6의 APV(적응적 프로브 볼륨)는 씬 지오메트리를 기반으로 라이트 프로브 배치를 자동화합니다. 따라서 라이트 프로브 그룹에서 수동으로 배치할 필요가 없습니다. APV는 오브젝트당 프로브와 달리 픽셀당 샘플링을 사용하기 때문에 조명 전환이 더 부드러워지고 게임 오브젝트 간 일관성이 향상됩니다.

APV는 픽셀당 적용되므로 수동으로 배치된 라이트 프로브로 인해 발생하는 여러 조명 문제가 방지됩니다. 다음은 라이트 프로브 그룹(왼쪽)과 APV(오른쪽)의 차이점을 보여 주는 비교 이미지입니다.



라이트 프로브 그룹과 APV 비교

라이트 프로브 그룹에서는 각 오브젝트가 하나의 보간된 프로브를 샘플링하기 때문에 각 부분의 조명이 일치하지 않는 것을 볼 수 있습니다. 반면에 APV는 픽셀당 샘플링을 사용하기 때문에 조명 전환이 더 매끄럽습니다.

아래는 라이트 프로브 그룹과 APV의 기능을 비교한 표입니다.

기능	라이트 프로브 그룹	APV (적응적 프로브 볼륨)
프로브 선택	게임 오브젝트당	픽셀당
스트리밍으로 메모리 사용량 최적화	No	Yes
자동으로 프로브 배치	No	Yes
여러 베이킹 간 블렌딩	No	Yes
수동으로 프로브 배치	Yes	No

APV의 장점:

- **픽셀당 광원:** 전환이 더 부드럽고 오브젝트 간의 경계가 더 적습니다.
- **향상된 안개 정확도:** 볼류메트릭 안개의 조명 배리어이션이 더 정밀합니다.
- **자동 프로브 배치:** APV를 베이킹하면 씬의 복잡도에 따라 프로브 밀도가 조절되어 주요 영역에서 디테일이 더 세밀해집니다.
- **다중 씬 지원:** **베이킹 세트**를 사용하면 여러 개의 씬을 함께 베이킹할 수 있습니다.
- **스트리밍 모드:** 이 기능은 대규모 오픈 월드의 메모리 사용량을 최적화합니다.
- **스카이 오클루전 업데이트:** APV는 런타임에 하늘의 조명 변화에 맞춰 동적으로 조정할 수 있습니다.
- **디버깅 툴:** 렌더링 디버거에 프로브 레이아웃과 밀도 수준이 시각화되어 문제 해결이 간편해집니다.

APV의 한계:

- **고정된 프로브 배치:** 라이트 프로브 위치를 수동으로 조정할 수 없어 벽이나 경계 부근에 아티팩트가 생길 수 있습니다.
- **라이트 프로브 그룹에서 변환 불가:** 기존 라이트 프로브 그룹을 APV로 변환할 수 없습니다.

다음 APV 볼륨 렌더링을 수동으로 배치한 라이트 프로브와 비교해 보세요. 많은 씬에서 라이트 프로브를 몇 초 이내에 배치할 수 있습니다.



프로브 볼륨은 라이트 프로브 렌더링을 개선합니다.

HDRP Quality 설정 또는 HDRP 에셋의 **Lighting > Light Probe System**에서 APV를 활성화할 수 있습니다. **Adaptive Probe Volumes**를 활성화하면 **Light Probe Groups**가 비활성화되며, 그 반대도 마찬가지입니다.

APV는 64개의 프로브(4 x 4 x 4)로 이루어진 ‘브릭’을 사용하여 씬 안에 자동으로 프로브를 배치합니다. APV는 씬의 지오메트리 밀도에 맞춰 적응하는 볼륨 기반 시스템을 사용합니다.

프로브 사이의 간격은 1, 3, 9 또는 27유닛으로 설정할 수 있으며, 이에 따라 브릭의 전체 크기가 달라집니다. 지오메트리 밀도가 높은 영역에서는 HDRP가 브릭을 더욱 빽빽하게 구성하므로 고해상도 조명 데이터를 얻을 수 있습니다. 밀도가 낮은 영역에서는 프로브 사이의 간격이 늘어납니다. 이를 통해 리소스를 필요한 영역에 집중적으로 사용할 수 있습니다.

다음은 베이킹된 프로브 볼륨에서 개별 프로브를 시각화한 모습입니다.



APV에서 라이트 프로브 사이의 간격을 여러 값으로 설정할 수 있습니다.

스폰자 궁전이란?

16세기 크로아티아 궁전을 기반으로 3D 컴퓨터 그래픽스 모델인 스폰자 아트리움이 제작되었습니다. 이 모델은 전역 조명 렌더링의 참조 모델로 널리 사용되고 있습니다. 유니티에서는 이 모델을 HDRP로 리마스터했으며, 이 GitHub 저장소에서 받을 수 있습니다.

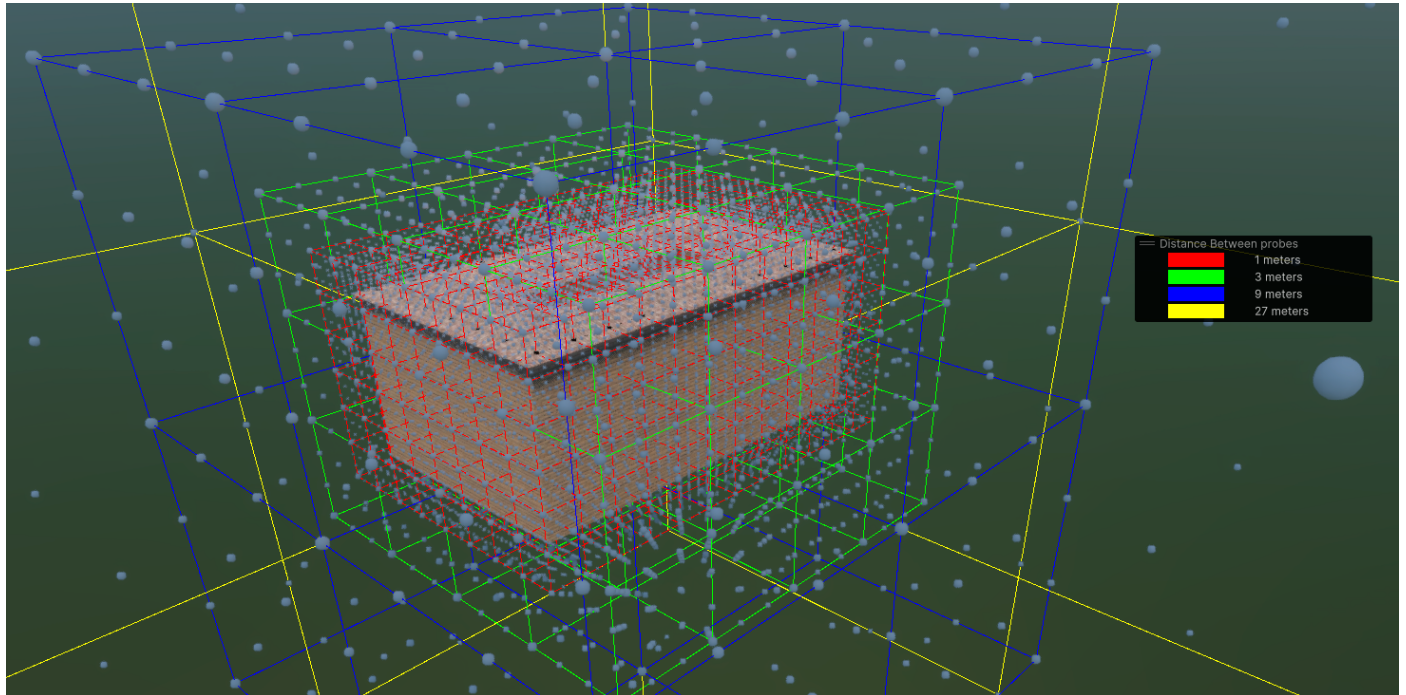
적응적 프로브 볼륨 사용

계층 구조 창을 오른쪽 클릭하고 **Light > Adaptive Probe Volume**(APV)을 선택합니다.

Mode를 **Global**로 설정하고 기본값을 사용하거나 특정 **Subdivision Override**를 선택합니다. **Override Probe Spacing**에서 새 값 범위를 선택합니다.

Bake Probe Volumes를 눌러 볼륨을 베이킹합니다. 프로브가 지오메트리에 따라 썬에 배치되며, 지오메트리가 많은 영역에서는 더 높은 밀도로 배치됩니다.

라이트 프로브 자체를 보려면 **Analysis > Rendering Debugger**를 엽니다. **Probe Volumes**를 선택하고 **Display Probes**를 활성화합니다. 다른 해상도를 보려면 **Display Bricks**를 선택합니다.



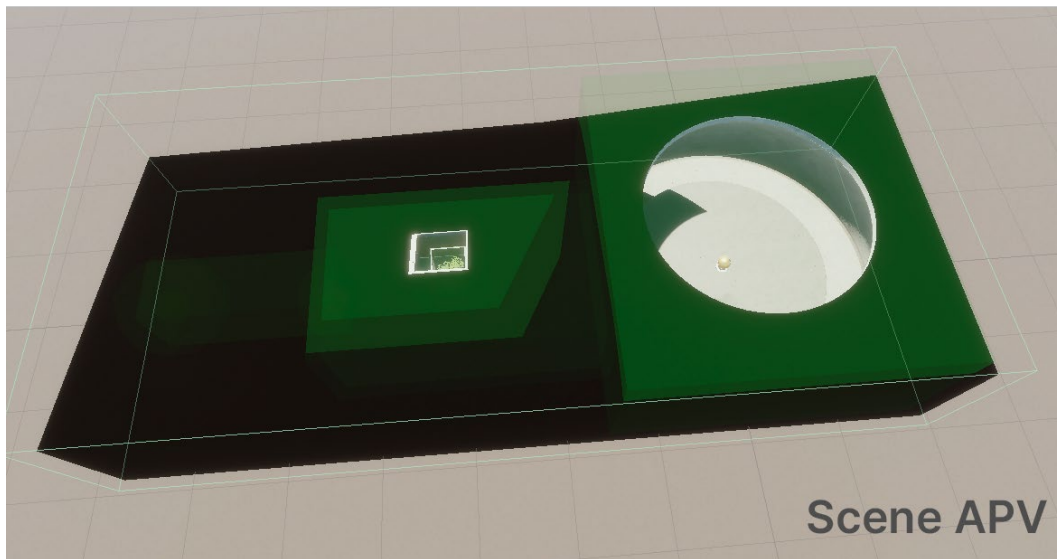
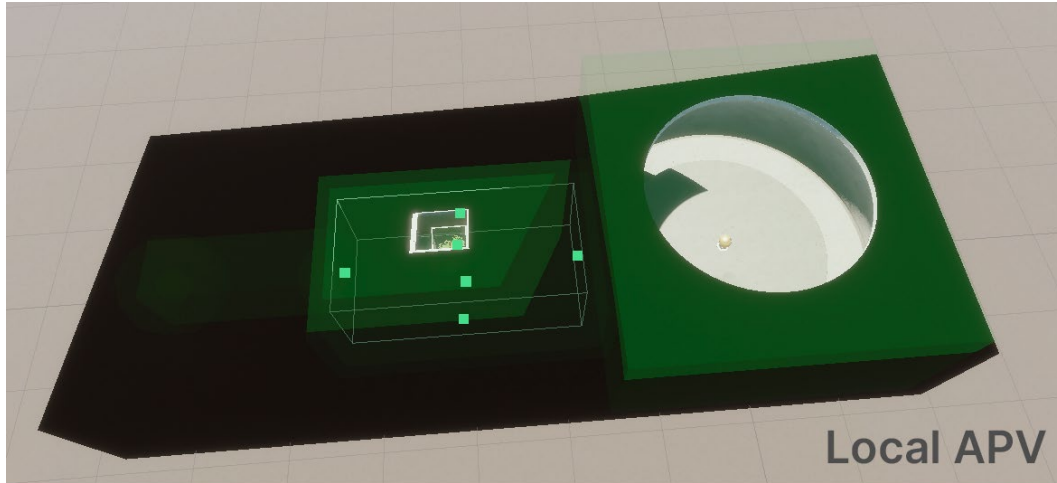
렌더링 디버거를 사용하여 APV를 봅니다.

APV 내에 있는 라이트 프로브의 개별 위치는 수동으로 조정할 수 없습니다. 따라서 프로브가 벽 또는 조명 경계에 꼭 맞게 정렬되지 않을 수 있습니다. 이에 대한 대안으로, 서로 다른 구획을 갖는 여러 개의 APV를 사용하여 프로브 배치와 밀도를 세밀하게 제어할 수 있습니다.

HDRP Sample 템플릿 썬도 이와 비슷한 설정을 사용합니다. 썬에는 2개의 프로브 볼륨이 있습니다. APV는 다음과 같은 모드를 사용하여 HDRP가 볼륨의 크기를 정하는 방식과 어느 렌더러가 영향을 미치는지를 제어합니다.

- **Global:** 베이킹 세트에 또는 Contribute Global Illumination이라고 표시된 썬에 있는 모든 렌더러를 포함하도록 APV의 크기가 자동으로 조정됩니다. 사용자가 조명을 저장하거나 재생성할 때마다 HDRP가 이 볼륨 크기를 다시 계산합니다.
- **Scene:** 현재 썬에 있는 모든 렌더러를 포함하도록 APV의 크기가 자동으로 조정됩니다.
- **Local:** APV의 크기가 자동으로 조정되지 않습니다. Size를 사용하여 볼륨을 수동으로 설정하세요.

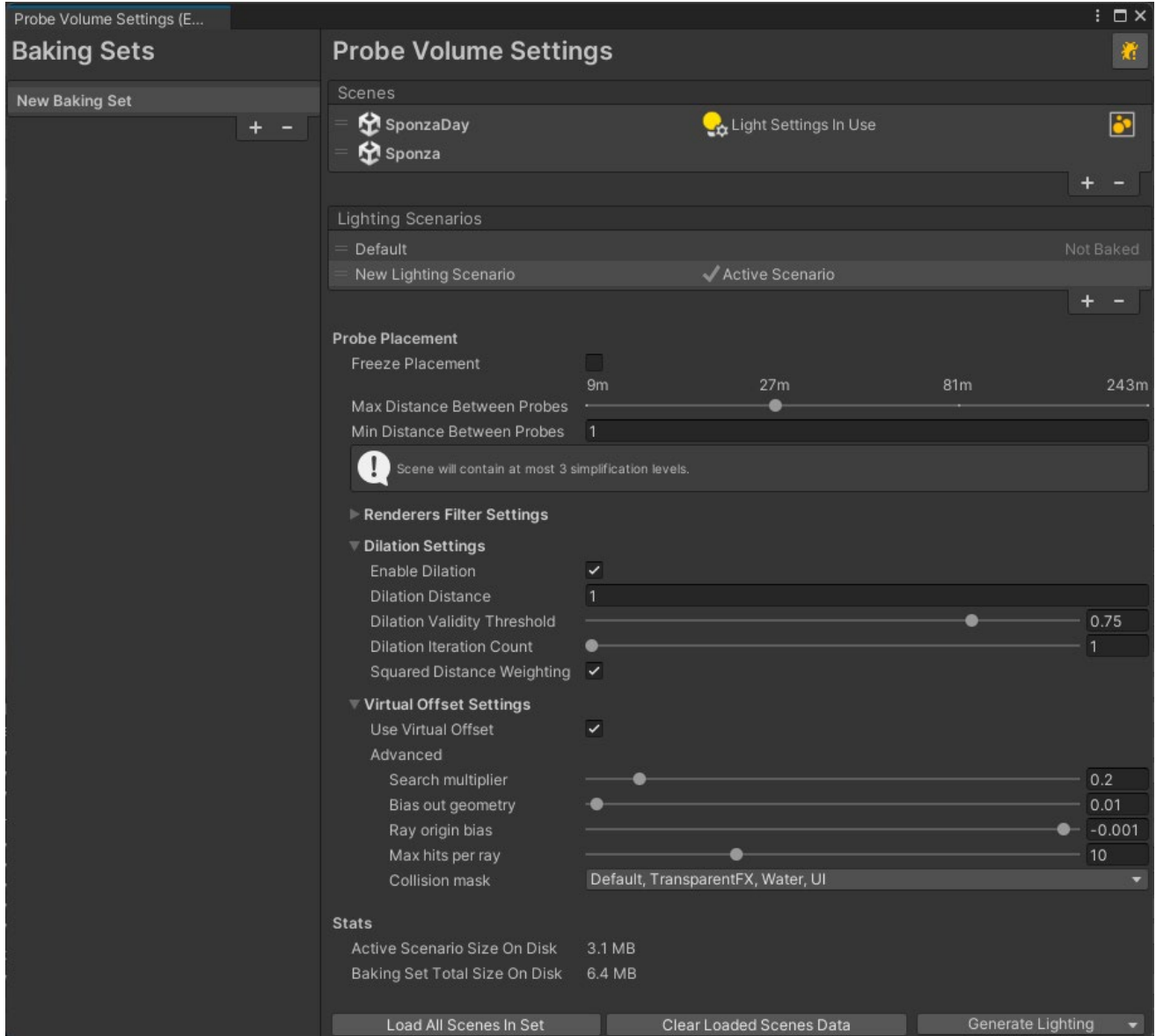
Sample 템플릿에서, 레벨 전체를 포함하는 APV는 **Mode**가 **Scene**으로 설정되어 있고, 두 번째 공간을 포함하는 APV는 **Mode**가 **Local**로 설정되어 있습니다.



씬에서 여러 개의 APV를 사용하여 세밀하게 제어하세요.

다음 중 하나에서 프로브 볼륨을 베이킹합니다.

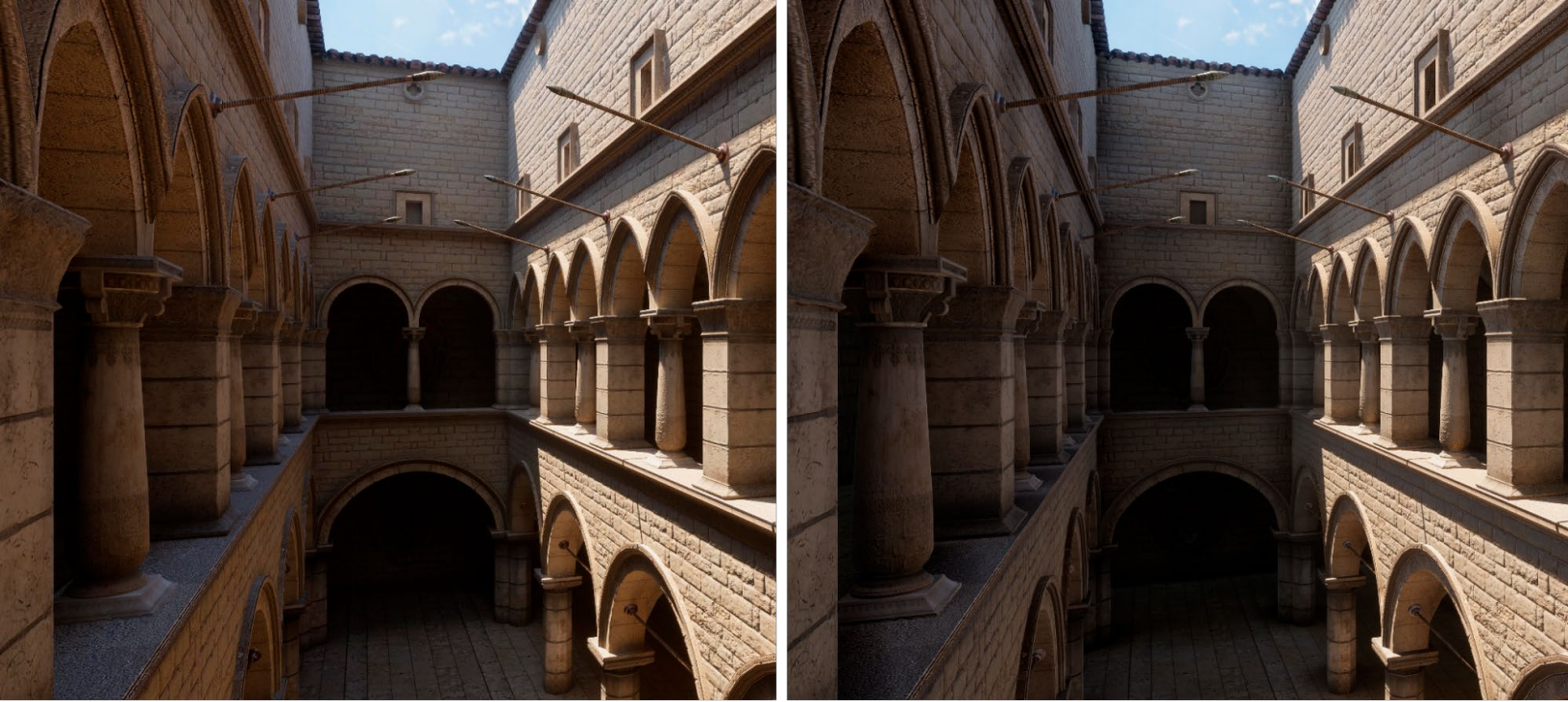
- **Adaptive Probe Volume 컴포넌트:** **Bake Probe Volumes**를 선택하여 특정 게임 오브젝트의 APV만 베이킹합니다.
- **Lighting 창:** **Generate Lighting**을 선택하여 APV를 반사 프로브 및 다른 베이킹된 광원과 함께 베이킹합니다.



프로브 볼륨 설정

완전히 APV로 전환하고 라이트맵을 아예 없애면 모든 씬 오브젝트에 하나의 일관된 조명 시스템을 사용하여 워크플로를 간소화할 수 있는 경우도 있습니다.

HDRP는 베이킹된 데이터를 여러 부분으로 분할하여 효율을 높입니다. 여러 조명 시나리오를 사용할 때 베이킹 간 프로브 배치와 지오메트리가 동일하게 유지되는 한 프로브 볼륨이 베이킹된 데이터를 중복하여 디스크에 저장하지 않습니다.



베이킹된 라이트맵(왼쪽)과 APV만 사용한 결과(오른쪽) 비교

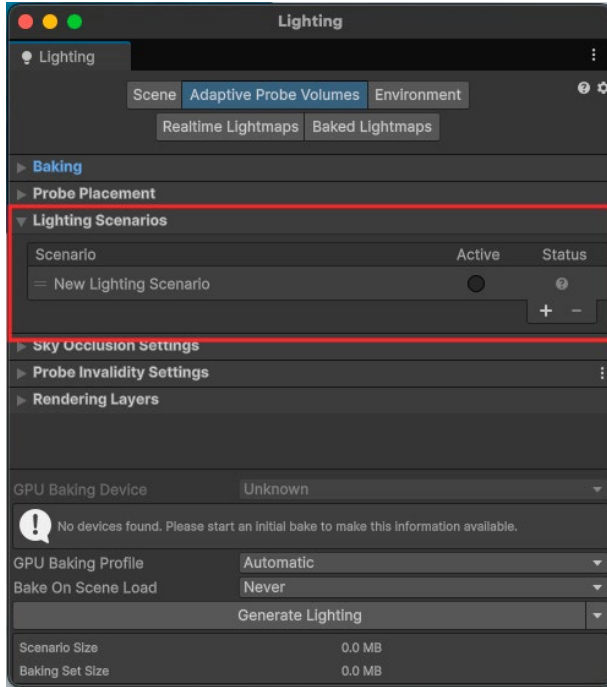
프로브 볼륨이 베이킹된 라이트매핑만큼 섬세한 차이를 모두 캡처하지는 못할 수 있지만, [스크린 공간 앰비언트 오클루전](#)과 같은 실시간 효과와 함께 사용하면 디테일을 더할 수 있습니다. 이렇게 하면 라이트맵 없이 간접 조명을 사용할 수 있습니다.

프로브 볼륨 설정 및 프로퍼티의 전체 목록을 참고해 보세요. 또한 [씬에 베이킹된 GI와 APV를 적용하는 방법](#)을 알아보려면 [Unity에서 조명 환경을 구현하는 4가지 기법](#)을 살펴보세요.

조명 시나리오 에셋

APV는 [조명 시나리오](#) 에셋을 사용하여 간접 조명 데이터를 전환할 수 있습니다. 각 조명 시나리오에는 베이킹된 조명 상태가 저장되어 있습니다. 다양한 조명 설정을 서로 다른 조명 시나리오로 베이킹한 다음, HDRP가 런타임에 사용할 조명 시나리오를 변경할 수 있습니다. 조명 시나리오는 씬 또는 베이킹 세트에 매칭될 수 있습니다.

예를 들어 낮 조명 시나리오와 밤 조명 시나리오를 생성할 수 있습니다. 런타임에 스크립트를 사용하여 두 시나리오를 전환하거나 블렌딩할 수 있습니다. 에디터에서 렌더링 디버거를 사용하여 효과를 미리 볼 수 있습니다.



Lighting 창에서 조명 시나리오를 적용합니다.

조명을 바로 전환하는 대신 **시나리오 블렌딩**을 사용하여 두 조명 시나리오를 점진적으로 전환하면 더 자연스러운 변화를 구현하게 됩니다. 이렇게 하면 급작스러운 전환을 방지할 수 있으며, 더 몰입감 있는 낮-밤 사이클이나 조명 상태 변화를 구현할 수 있습니다.

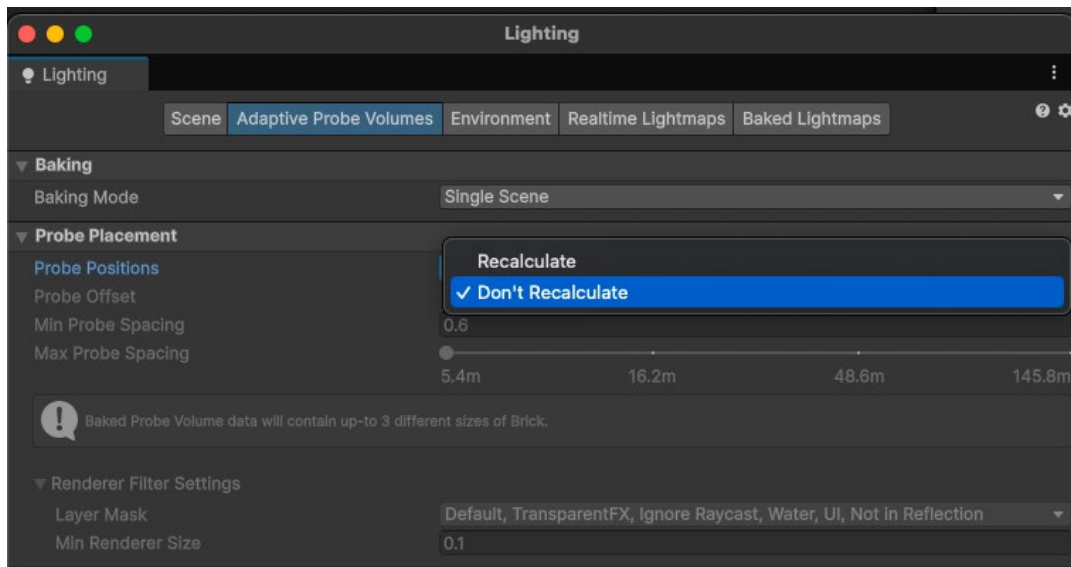


하나의 환경에 여러 조명 시나리오를 사용할 수 있습니다.

조명 시나리오 에셋을 사용하려면 활성 HDRP 에셋에서 **Lighting > Light Probe Lighting > Lighting**

Scenarios를 활성화합니다. 그런 다음 베이킹 결과를 저장하기 위한 새로운 조명 시나리오 에셋을 생성할 수 있습니다.

1. **Lighting** 창에서 **Adaptive Probe Volumes** 패널을 엽니다.
2. **Lighting Scenarios** 섹션에서 **추가(+)** 버튼을 선택하여 조명 시나리오 에셋을 추가합니다.
3. **Lighting** 창의 **Adaptive Probe Volume** 탭에서 **Probe Positions**를 **Don't Recalculate**로 설정해야 합니다. 이렇게 설정해야만 Unity에서 프로브 위치를 바꾸는 일 없이 조명만 다시 베이킹합니다. 그러지 않을 경우에는 이전 베이킹 시나리오가 무효화될 수 있습니다.



조명 시나리오를 사용할 때 프로브 위치를 다시 계산하지 않도록 설정합니다.

목록에서 원하는 조명 시나리오를 선택하여 활성화하고 **Generate Lighting**을 선택합니다. HDRP가 베이킹 결과를 활성화된 조명 시나리오에 저장합니다.

Generate Lighting 옆의 드롭다운 버튼을 사용하여 라이트맵을 사용하지 않는 경우에만 **Bake Probe Volumes**가 적용되도록 선택합니다.

[ProbeReferenceVolume](#) API를 사용하여 런타임에 사용할 조명 시나리오를 설정하고 [BlendLightingScenario](#) API를 사용하여 블렌딩합니다.

렌더링 디버거(**Window > Analysis > Rendering Debugger**)를 사용하여 조명 시나리오 간의 전환을 미리 볼 수도 있습니다. Scenario Blend Target을 원하는 조명 시나리오로 설정하고 시나리오 블렌드 인자를 사용하여 블렌딩 효과를 확인합니다.

시나리오 블렌드 인자는 두 베이킹된 광원 상태 간의 보간을 제어합니다. 시나리오 블렌드 인자는 하루 동안의 시간 변화에 따른 동적인 효과, 실내-실외 전환, 동일한 씬 내의 서로 다른 조명 분위기에 사용할 수 있습니다. 시나리오 블렌딩을 사용하면 조명 시나리오가 변경될 때 이질감 없이 부드럽고 점진적인 전환이 이루어집니다.

Time Ghost: Environment 에셋에서 시나리오 블렌딩의 예시를 볼 수 있습니다. Time Ghost: Environment에서는 MonoBehaviour가 여러 조명 시나리오를 블렌딩하여 하루 동안의 시간 변화에 따른 부드러운 전환을 시뮬레이션합니다.

조명 시나리오 관리자

런타임에 활성 조명 시나리오를 변경하면 HDRP는 라이트 프로브의 간접 조명 데이터만 변경합니다. 지오메트리를 이동하거나 광원 또는 직접 조명을 수정하려면 스크립트를 사용해야 할 수 있습니다.

[TimeGhost: Environment](#) 샘플의 ScenarioManager 스크립트를 살펴보세요.

APV 문제 해결

APV는 프로브를 자동으로 그리드 형태로 배치하므로 배치로 인해 렌더링 오류가 발생하는 경우가 있습니다.

프로브가 지오메트리 내에 있고 샘플링 광선이 빛을 받지 않는 후면에 도달하여 정확한 조명을 받지 못하는 경우 유효하지 않은 프로브가 발생합니다. 이로 인해 밝아야 하는 영역이 어두워지거나, 어두워야 하는 영역이 밝아질 수 있습니다.

이 예시에서는 3D 샘플 씬 바닥 아래에 배치된 유효하지 않은 프로브로 인해 캡슐 밑부분에 어두운 아티팩트가 발생했습니다.



유효하지 않은 프로브는 어두운 아티팩트를 유발합니다.

에디터에는 테크니컬 아티스트가 이러한 문제를 발견하고 해결할 수 있는 다양한 툴이 있습니다.

- **렌더링 디버거: Probes Volumes > Debug Probe Sampling**으로 이동하여 특정 픽셀에 어느 프로브가 영향을 주는지 살펴봅니다.
- **Probe Invalidity Settings: Lighting** 창의 **Adaptive Probe Volumes** 패널로 이동합니다. 이 설정을 사용하여 더 효과적인 샘플링이 이루어지도록 각 프로브의 캡처 포인트를 오프셋합니다.

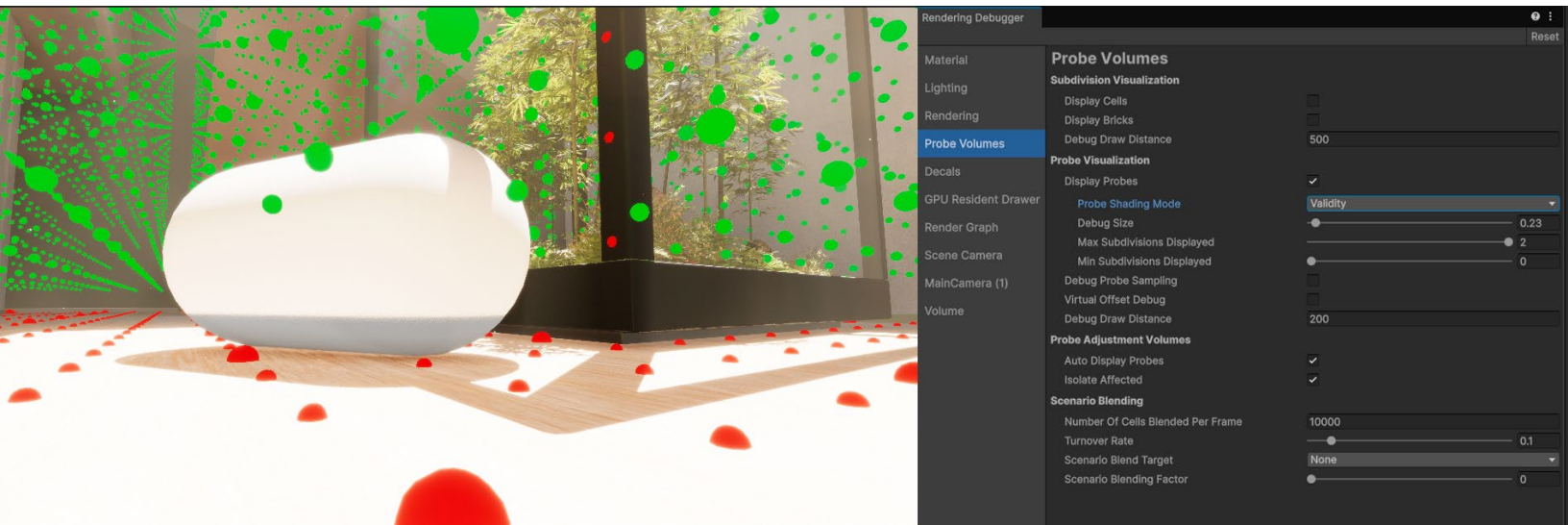
APV의 몇 가지 일반적인 문제와 해결 방법을 살펴보겠습니다.

유효하지 않은 프로브 해결하기

일반적으로 발생하는 대표적인 문제로 프로브가ジオ메트리 내에서 생성되는 경우를 들 수 있습니다. 오브젝트 내의 후면을 샘플링하는 프로브는 올바르지 않은 조명을 유발할 수 있으며 유효하지 않은 프로브로 간주됩니다.

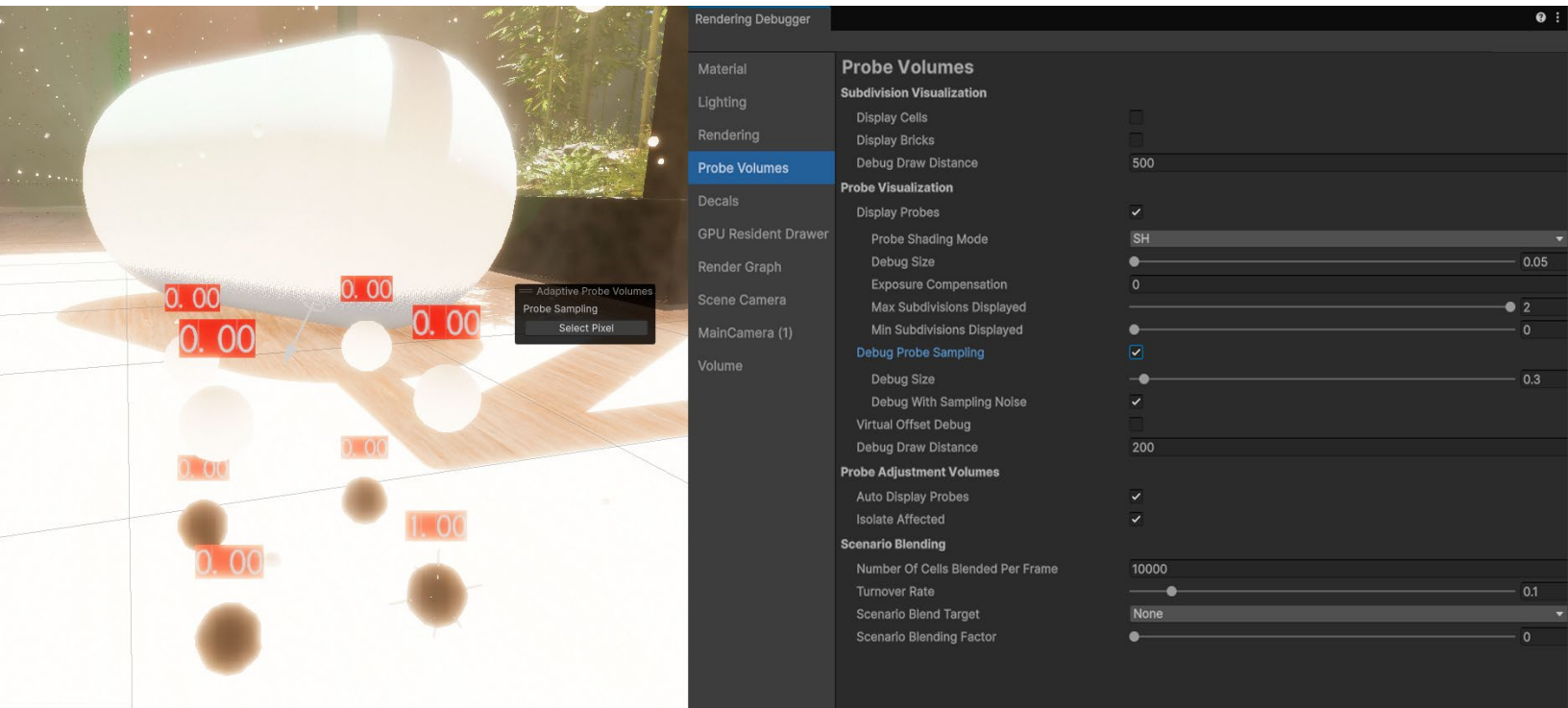
렌더링 디버거(Window > Analysis > Rendering Debugger > Probe Volumes)를 사용하여 문제를 해결합니다. **Display Probes** 옵션을 활성화하고 **Probe Shading Mode**를 **Validity**로 설정하여 유효하지 않은 프로브를 표시합니다.

위 예시에서는 이제 게임 뷰에 APV의 개별 프로브가 표시됩니다. 유효한 프로브는 녹색으로, 유효하지 않은 프로브는 빨간색으로 표시됩니다.



렌더링 디버거는 유효하지 않은 프로브를 빨간색으로 표시합니다.

특정 아티팩트를 유발하는 요인을 가려내려면 **Debug Probe Sampling**을 활성화합니다. 씬 뷰에서 **Select Probes**를 선택하고 캡슐의 밑면을 클릭합니다. 이 디버거 뷰는 해당 픽셀에 어느 프로브가 영향을 주는지 보여 줍니다.

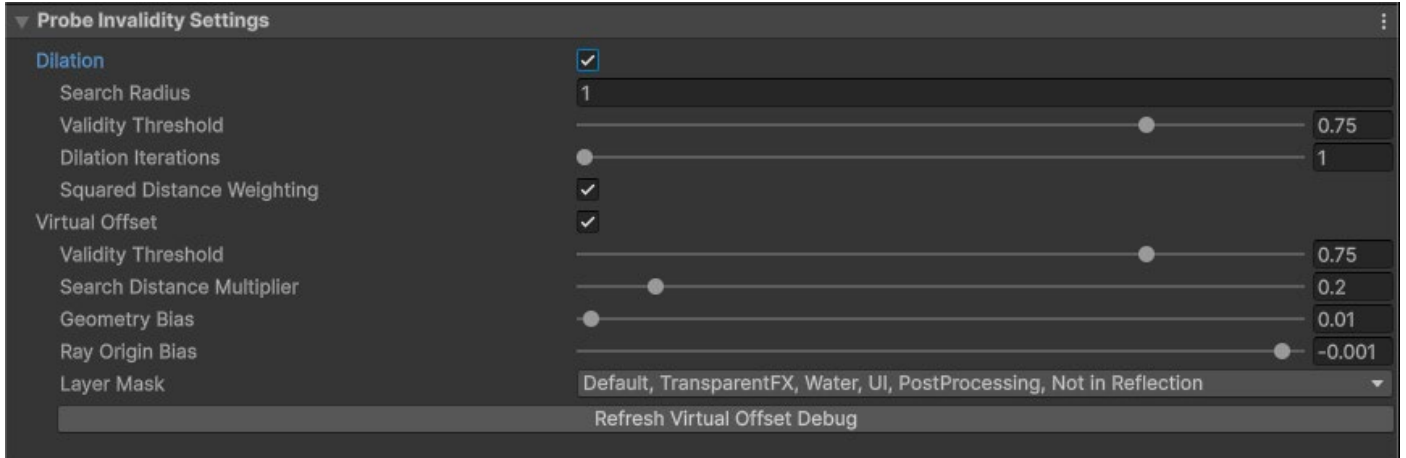


Debug Probe Sampling 옵션을 선택하면 어느 프로브가 문제를 유발하는지 볼 수 있습니다.

Probe Invalidation Settings(Lighting 창 > Adaptive Probe Volumes 패널)를 조정하여 유효하지 않은 프로브로 유발된 어두운 영역을 수정합니다.

- **Virtual Offset**은 지오메트리 밖의 유효한 지점을 찾기 위해 광선을 투사하여 각 프로브의 캡처 위치를 이동합니다. **Validity Threshold**, **Search Distance Multiplier**, **Layer Mask**와 같은 설정을 조정하여 프로브 배치를 개선해 봅니다. **Geometry Bias**는 프로브를 가까운 표면으로부터 멀리 떨어뜨리고 **Ray Origin Bias**는 배치 정확도가 향상되도록 프로브 광선이 시작되는 지점을 조정합니다.
- **Dilation** 설정은 가까운 유효한 프로브에서 조명 데이터를 빌려 누락된 데이터에 채워 넣음으로써 유효하지 않은 프로브를 수정합니다. **Search Distance**, **Validity Threshold**, **Iterations**와 그 밖의 Dilation 설정으로 이 프로세스를 미세 조정하여 프로브가 얼마나 엄격하게 검증되는지 제어하세요. **Squared Distance Weighting**은 빌린 조명 데이터를 더 자연스럽게 블렌딩합니다.

어두운 부분이 사라질 때까지 설정을 조정해 보며 프로브 볼륨을 다시 베이킹하세요. Virtual Offset과 Dilation은 베이킹 프로세스에만 영향을 주며 런타임 성능에는 영향을 주지 않습니다.



Probe Invalidation Settings를 사용하여 유효하지 않은 프로브로 인해 발생한 어두운 영역을 수정합니다.

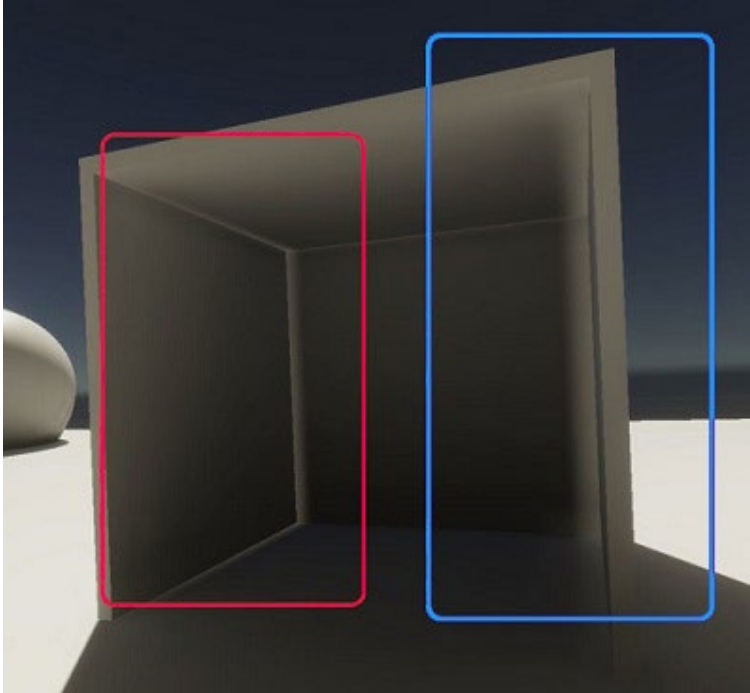


Virtual Offset 설정과 Dilation 설정을 조정한 후 아티팩트가 제거되었습니다.

빛 번짐 현상 해결하기

‘빛 번짐 현상’은 유효하지 않은 프로브로 인한 어두운 영역과 마찬가지로, 지오메트리가 해당 지오메트리에 보이지 않는 라이트 프로브로부터 빛을 받을 때(예: 라이트 프로브가 벽 너머에 있을 때) 발생할 수 있는 결함입니다.

이 현상은 환경의 APV 밀도와 벽 두께가 일치하지 않아 보간이 이루어지고, 이로 인해 바깥쪽 조명이 불투명한 벽을 통과하는 것처럼 보이는 경우에 발생합니다.



빛 번짐 현상은 결함으로 간주됩니다.

빛 번짐 현상은 주로 벽이나 천장의 모서리에서 너무 밝거나 너무 어두운 영역으로 나타납니다. APV는 일반적인 그리드 형태의 라이트 프로브를 사용하므로 라이트 프로브가 벽의 형태를 따르지 않거나 서로 다른 조명 영역 간의 경계에 있을 수 있습니다.

빛 번짐 현상 해결에 사용할 수 있는 전략은 다음과 같습니다.

- 프로브 배치가 개선되도록 벽 두께를 늘립니다. 벽 두께가 로컬 [브릭](#)에 속한 프로브 간 거리와 더 가까워지도록 조절합니다.
- Adaptive Probe Volumes 옵션 오버라이드 (아래 참조)를 사용하여 샘플링 위치를 조정합니다.
- 오브젝트가 올바르게 얹은 레이어의 프로브를 사용하지 않도록 Rendering Layer Masks(아래 참조)를 활성화합니다.
- Baking Set 프로퍼티를 수정하여 프로브 간격과 검색 거리를 최적화합니다.

Probe Adjustment Volume 사용

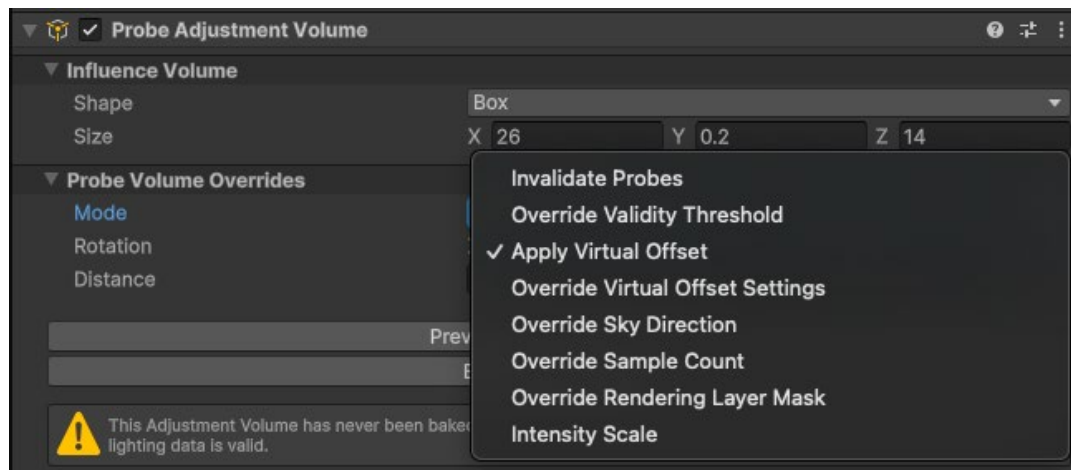
볼륨 시스템에는 **Probe Adjustment Volume** 옵션 오버라이드가 있습니다. 이 오버라이드는 사용자가 씬의 특정 영역에서 프로브 기반 조명을 조정할 수 있도록 라이트 프로브와 APV에 대한 국지화된 제어를 제공합니다.

Influence Volume 설정을 사용하여 영향을 줄 **Box** 또는 **Sphere**를 정의합니다. 그런 다음 **Mode**를 선택하여 강도, 샘플링, 하늘에 미치는 영향 또는 프로브 유효성을 조정합니다.

- **Invalidate Probes:** 이 모드에서는 프로브를 유효하지 않은 것으로 표시하여 프로브가 조명 계산에 기여하지 않도록 합니다.

- **Override Validity Threshold:** 프로브가 검증되는 방식을 변경하여 유효하지 않은 프로브로 인한 아티팩트를 줄입니다.
- **Apply Virtual Offset:** Lighting Window의 Rotation과 Distance를 로컬 가상 오프셋으로 오버라이드합니다.
- **Override Virtual Offset Settings:** 이 모드는 Lighting Window 설정의 **Validity Threshold**, **Geometry Bias**(프로브를 가까운 표면으로부터 멀리 떨어뜨림), **Ray Origin Bias**(프로브 광선이 시작되는 지점을 조정함)를 오버라이드합니다.
- **Intensity Scale:** 이 모드는 볼륨 내의 밝기를 조정합니다.
- **Override Sky Direction:** 프로브가 특정 하늘 방향을 사용하도록 강제합니다.
- **Override Layer Mask:** 어느 프로브가 영향을 받는지 선택할 때 프로브가 다른 레이어 마스크를 사용하도록 합니다.

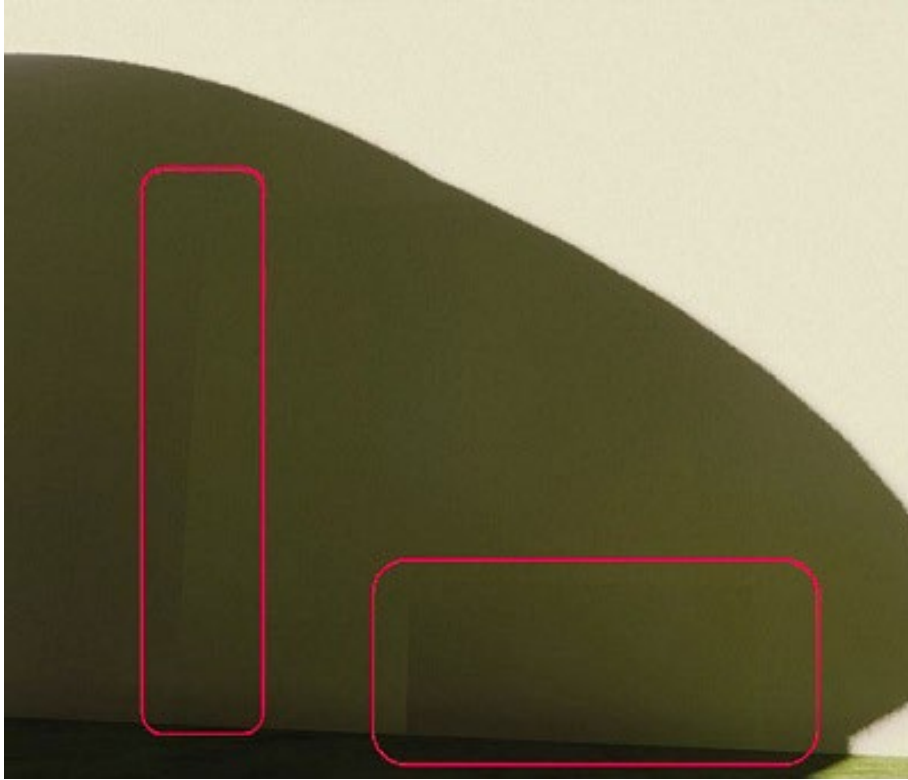
볼륨은 볼륨 근처 또는 내부에 카메라가 있는 경우에만 씬에 영향을 줍니다. **Probe Volumes Options** 설정에 대한 자세한 내용은 [볼륨 이해하기](#)와 [Probe Volumes Options Override 레퍼런스](#)를 참조하세요.



이 오버라이드는 라이트 프로브와 APV에 대한 국지화된 제어를 제공합니다.

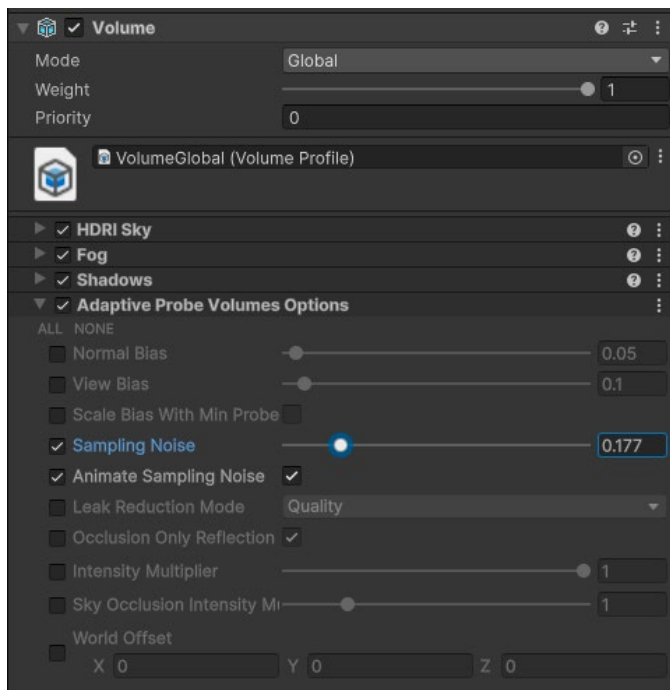
경계 해결하기

경계는 인접한 두 프로브 브릭의 밀도가 달라 급작스러운 조명 전환이 생성되는 경우에 나타납니다. HDRP가 경계를 자동으로 수정하긴 하나, 계속 지속되는 경우 Adaptive Probe Volumes Options 오버라이드에서 **Sampling Noise**를 활성화하여 전환을 부드럽게 만들 수 있습니다.



두 프로브 브릭 사이의 경계 예시.

Baking Set 프로퍼티와 프로브 밀도를 조절하면 결과가 더 개선될 수 있습니다. 대부분의 APV 문제는 베이크 중에 나타나니, 씬을 가장 효율적으로 렌더링하려면 조명을 충분히 조정해 다음을 확인하세요.



Sampling Noise를 조정하여 경계를 해결합니다.

렌더링 레이어 마스크

APV의 문제를 해결하는 또 다른 방법은 렌더링 레이어 마스크를 사용하는 것입니다. 렌더링 레이어 마스크는 오브젝트가 관련 프로브에서 조명을 샘플링하도록 하는 필터로 작동하여 빛 번짐 현상을 줄여 줍니다. APV는 최대 4개의 렌더링 레이어 마스크를 지원합니다.

Unity는 광원 베이크 중에 가까운 오브젝트를 기반으로 각 프로브에 하나의 렌더링 레이어 마스크를 자동으로 할당합니다. 특정 렌더링 레이어 마스크(예: 'Interior')를 갖는 오브젝트 가까이에 있는 프로브는 해당 레이어를 상속받습니다.

런타임에 오브젝트는 자신과 동일한 렌더링 레이어 마스크를 갖는 프로브에서만 조명을 샘플링합니다. 일치하는 마스크가 없으면 주변에 있는 모든 유효한 프로브에서 샘플링합니다.

빛 번짐 현상의 가장 흔한 원인은 서로 크게 다른 조명을 갖는 내부 프로브와 외부 프로브 간에 픽셀이 블렌딩되는 경우입니다. 이를 방지하려면 내부용 렌더링 레이어 마스크와 외부용 렌더링 레이어 마스크를 각각 만들면 됩니다.

이 기능을 사용하려면 HDRP 에셋에서 **Light Layers**가 활성화되어 있어야 합니다.

APV에 대해 렌더링 레이어 마스크를 활성화하는 방법은 다음과 같습니다.

1. **Mesh Renderer** 컴포넌트에 있는 렌더러에 **Rendering Layer Masks**를 할당합니다.
2. **Lighting** 창을 열고 **Adaptive Probe Volumes > Rendering Layer Masks**로 이동합니다.
3. 목록에 최대 4개의 Rendering Layer Masks를 할당합니다. 베이크 중에, 가까운 오브젝트를 기반으로 마스크에 라이트 프로브가 할당됩니다.

HDRP에서는 렌더링 디버거를 사용하여 오브젝트에 어느 레이어가 할당되었는지 볼 수 있습니다.

- Material 탭으로 이동합니다.
- **Material** 필드를 **Common > Rendering Layers**로 설정합니다.

APV의 스카이 오클루전

APV의 스카이 오클루전은 그늘지거나 가려진 영역(예: 터널, 방, 돌출부 밑)에서 하늘 광원 기여도를 줄임으로써 간접 조명을 동적으로 조정합니다. 이로 인해 더 사실적인 간접 조명이 생성되며, 하늘의 영향을 덜 받아야 하는 영역이 과하게 밝아지지 않도록 방지할 수 있습니다.

앰비언트 프로브는 런타임에 업데이트될 수 있기 때문에 낮에서 밤으로의 부드러운 전환 같은 실시간 조명 전환이 가능합니다. 따라서 APV만 사용하여 낮-밤 사이클을 시뮬레이션할 수 있습니다.

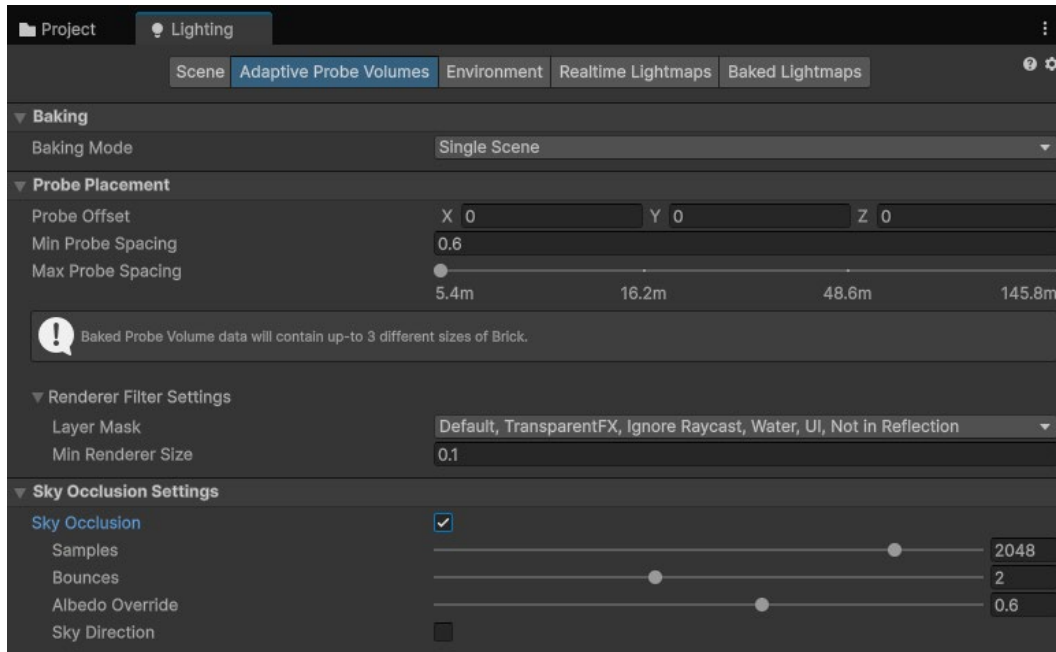
스카이 오클루전 활성화

스카이 오클루전을 활성화하는 방법은 다음과 같습니다.

- Lighting 창에서 **Progressive GPU Lightmapper**를 활성화합니다. 프로그레시브 CPU를 사용하는 경우에는 이 기능이 지원되지 않습니다.
- Adaptive Probe Volumes 패널에서 **Sky Occlusion**을 활성화하고 볼륨을 다시 베이킹합니다.

Sky Occlusion을 활성화하면 Unity가 각 프로브에 프로브가 받은 간접 하늘 광원을 나타내는 정적 스카이 오클루전 값을 추가로 베이킹합니다. Unity는 런타임에 이를 앰비언트 프로브의 하늘 색상과 결합하며, 이 하늘 색상은 하늘이 변경되면 동적으로 업데이트됩니다.

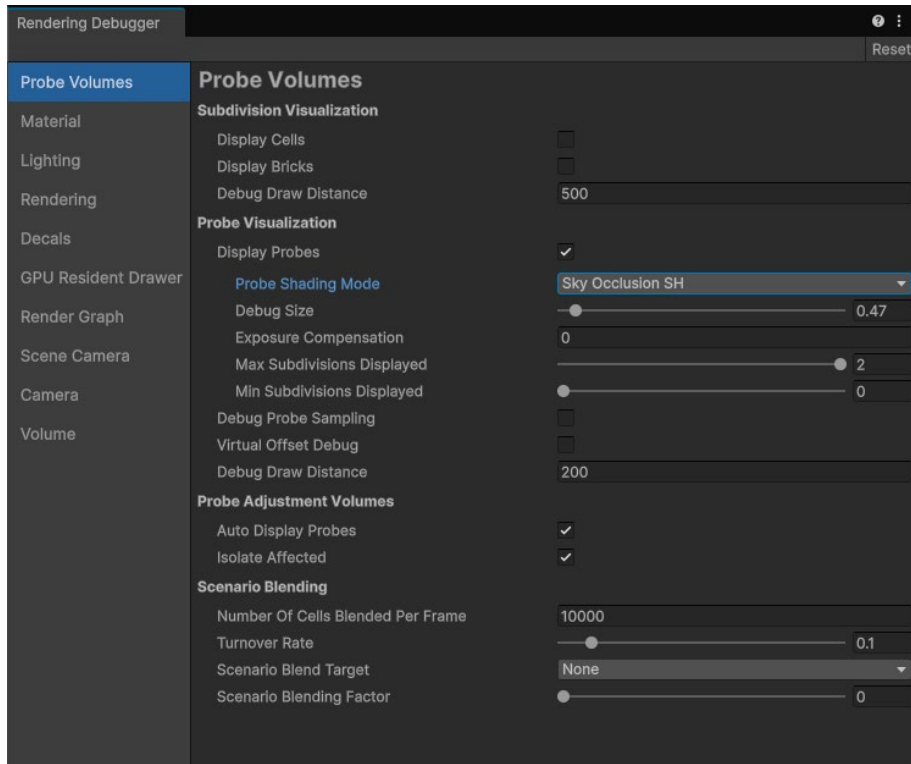
앰비언트 프로브가 런타임에 업데이트되도록 하려면 **Volume Profile Asset**을 열고 **Visual Environment** > **Sky** 섹션으로 이동한 다음 **Ambient Mode**를 **Dynamic**으로 설정합니다.



Lighting 창에서 Sky Occlusion 옵션을 활성화합니다.

스카이 오클루전 디버깅

Rendering Debugger에서 스카이 오클루전이 APV 조명에 영향을 주는 방식을 볼 수 있습니다. **Probe Volumes** 섹션에서 **Display Probes**를 활성화하고 **Probe Shading Mode**를 **Sky Occlusion SH**로 설정합니다.



Rendering Debugger에서 스카이 오클루전을 디버깅합니다.

Sky Occlusion이 활성화된 왼쪽 이미지와 비활성화된 오른쪽 이미지를 비교해 보세요. Sky Occlusion이 활성화된 경우, 열린 공간에 있는 프로브는 더 밝게 보이고 씬 지오메트리로 막힌 프로브는 어두워 보입니다. 즉, 간접적인 하늘 조명이 적절하게 줄어듭니다. Sky Occlusion이 비활성화된 경우, 모든 프로브가 균일한 하늘 조명을 받습니다. 이로 인해 그늘진 영역이 비정상적으로 밝아 보입니다.



Sky Occlusion 설정이 활성화된 왼쪽 이미지와 비활성화된 오른쪽 이미지

이 차이는 하늘이 가려진 영역에서 가장 뚜렷하게 볼 수 있습니다. 스카이 오클루전이 베이킹되면 씬 조명이 앰비언트 프로브의 변경에 따라 업데이트됩니다. HDRP에서는 Color 또는 Gradient 모드를 사용할 때만 앰비언트 프로브가 실시간으로 업데이트되며, Skybox 모드에서는 실시간으로 업데이트되지 않습니다.

하늘 방향

Unity는 앰비언트 프로브를 샘플링할 때 기본적으로 오브젝트의 표면 노멀을 하늘 방향으로 사용합니다. 하지만 하늘 광원이 다른 오브젝트에 반사되는 밀폐된 영역에서는 표면 노멀이 하늘 방향이 아닐 수 있습니다.

Unity가 각 프로브별로 정밀한 하늘 조명 방향을 계산하고 저장할 수 있도록 **Adaptive Probe Volumes** 패널에서 **Sky Direction**을 활성화합니다. 그러면 프로브에 하늘까지의 직접 가시선이 없는 영역이나 광원이 특정 방향에서(예: 창문을 통해) 오는 경우에 반사되는 빛이 고려될 수 있어, 베이킹 시간과 메모리가 추가로 사용되는 대신 조명 정확도가 개선됩니다.

APV 데이터 로딩 최적화

런타임에 APV 데이터 로딩을 최적화하려면 APV 데이터를 스트리밍하거나 AssetBundles/Addressables에서 로딩할 수 있습니다.

APV 데이터 스트리밍

스트리밍하면 카메라가 이동함에 따라 반드시 필요한 데이터만 로딩하여 대규모 APV 조명을 사용할 수 있습니다. 그러면 메모리 사용량이 줄어들고 가용 CPU/GPU 메모리보다 큰 APV 데이터를 베이킹할 수 있습니다. HDRP는 StreamingAssets 폴더에서 데이터를 스트리밍하고 카메라의 뷰 절두체 내에 있는 셀만 로딩합니다.

APV 스트리밍을 활성화하는 방법은 다음과 같습니다.

1. **Edit > Project Settings > Quality > HDRP**로 이동합니다.
2. 원하는 **Quality Level**을 선택한 다음 **Lighting > Light Probe Lighting**을 펼칩니다.

이제 두 가지 유형의 스트리밍을 활성화할 수 있습니다.

- CPU 메모리에서 GPU 메모리로 스트리밍하려면 **Enable GPU Streaming**을 선택합니다.
- 디스크에서 CPU 메모리로 스트리밍하려면 **Enable Disk Streaming**을 선택합니다. **Enable GPU Streaming**을 먼저 활성화해야 합니다.

HDRP 에셋 설정에서 스트리밍 설정을 추가로 구성할 수 있습니다.

HDRP가 로드하고 사용하는 가장 작은 섹션 단위는 셀로, 이는 적응적 프로브 볼륨의 가장 큰 **브릭**과 같은 크기입니다. **라이트 프로브의 밀도를 조정**하여 APV 내 셀의 크기를 조절할 수 있습니다. 렌더링 디버거를 사용하여 APV 셀을 보고 스트리밍을 디버깅하세요.

AssetBundles 또는 Addressables에서 APV 데이터 로딩

런타임에 꼭 필요한 APV 데이터만 로딩하려면 Streaming Assets 대신 AssetBundles 또는 Addressables에 데이터를 저장합니다.

1. AssetBundle-Based APV Loading을 활성화하고, **Project Settings > Graphics > Pipeline Specific Settings > HDRP로 이동합니다.**
2. **Probe Volume Disable Streaming Assets**를 활성화합니다(디스크 스트리밍이 비활성화됩니다).
3. 조명을 베이킹합니다.
4. 씬을 AssetBundle에 추가합니다.

참고: Unity는 일부 씬이 로딩되지 않더라도 모든 베이킹 세트 조명 데이터를 메모리에 저장하기 때문에 Streaming Assets를 비활성화하면 메모리 사용량이 증가할 수 있습니다.

런타임에 라이트 프로브 베이킹

Unity 6에는 기존 방법에 비해 더 많은 제어와 효율성, 유연성을 제공하는 새로운 라이트 프로브 베이킹 API가 도입되었습니다. 이제 한 번에 베이킹되는 프로브 수를 제어하여 실행 시간과 메모리 사용량의 균형을 맞출 수 있습니다.

API에는 고성능 **GPU 가속 백엔드**가 있으며, 프로브 베이킹에 **OpenCL**을 사용합니다. 이 시스템은 베이킹에 앞서 씬에서 직접 필요한 조명 데이터를 추출하며, 절차적 지오메트리는 지원하지 않습니다.

자세한 내용은 다음 기술 자료 페이지를 참조하세요.

- [RadeonRaysContext](#)
- [RadeonRaysProbeIntegrator](#)
- [RadeonRaysProbePostProcessor](#)

적응적 프로브 볼륨에 대해 자세히 알아보려면 GDC 2023 세션 [적응형 프로브 볼륨을 활용한 효율적이고 강렬한 조명](#)을 확인하세요. [적응적 프로브 볼륨 기술 자료](#)도 참고하면 좋습니다.

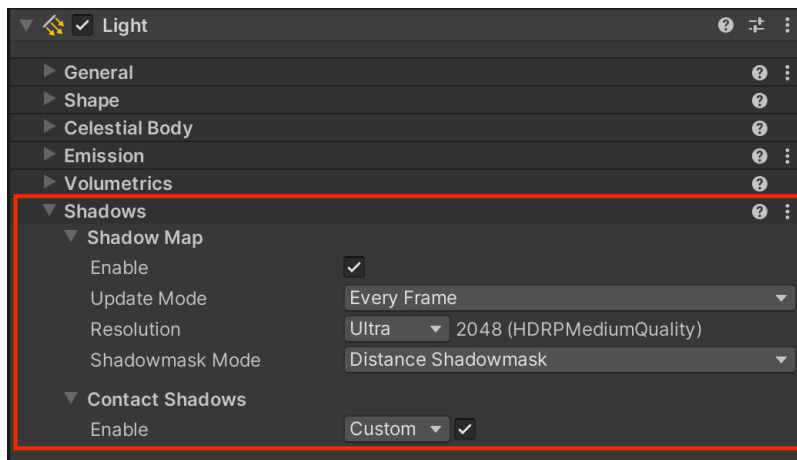
그림자

씬에 그림자를 적절히 배치하면 조명만큼 흥미롭게 환경을 조성하고 씬에 깊이와 입체감을 추가할 수 있습니다. HDRP에는 그림자를 조정하고 렌더링된 결과물이 맛있게 보이지 않도록 하는 다양한 기능이 있습니다.

새도우 맵

그림자는 텍스처가 광원의 시점에서 뎁스 정보를 저장하는 **새도우 매핑**이라는 기술을 사용해 렌더링됩니다.

Light 컴포넌트의 Shadows 하위 섹션에서 새도우 매핑의 **Update Mode** 옵션과 **Resolution** 값을 수정할 수 있습니다. 해상도가 높고 업데이트 주기가 짧을수록 리소스를 더 많이 사용합니다.



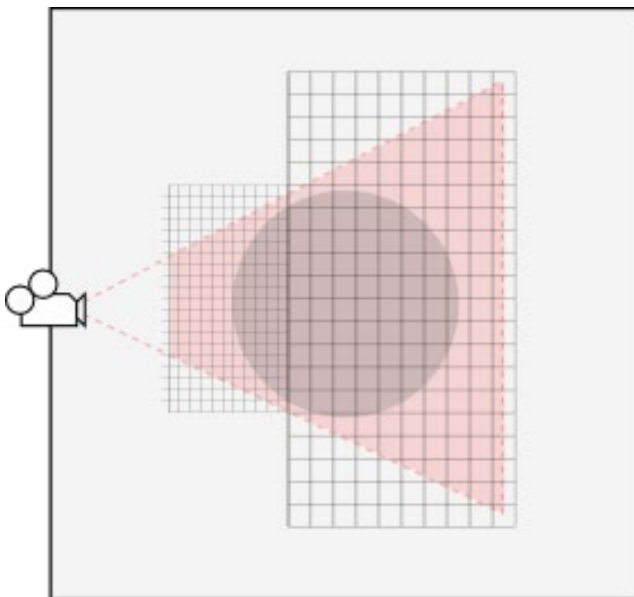
광원별 그림자 설정

Shadow Cascades

방향 광원의 경우, 새도우 맵이 씬의 많은 부분을 덮기 때문에 원근 앨리어싱이라는 문제가 발생할 수 있습니다. 카메라에 가까운 새도우 맵 픽셀은 멀리 있는 픽셀보다 들쭉날쭉하고 뭉툭해 보입니다.



그림자가 뭉툭해 보이는 원근 앨리어싱



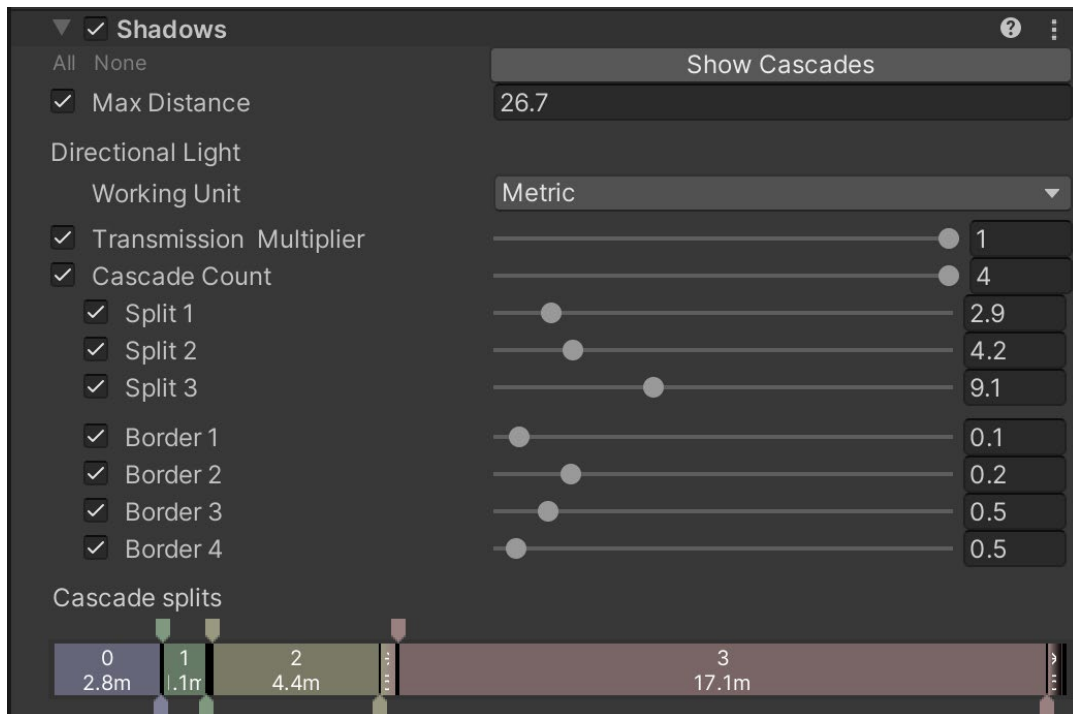
그림자 캐스케이드는 카메라 절두체를 각각 고유한 새도우 맵이 있는 영역으로 나눕니다.

Unity는 **캐스케이드드 새도우 맵**을 사용해 이 문제를 해결합니다. 캐스케이드드 새도우 맵은 카메라 절두체를 각각의 고유한 새도우 맵이 있는 영역으로 분할합니다. 그러면 원근 앨리어싱 현상이 줄어듭니다.



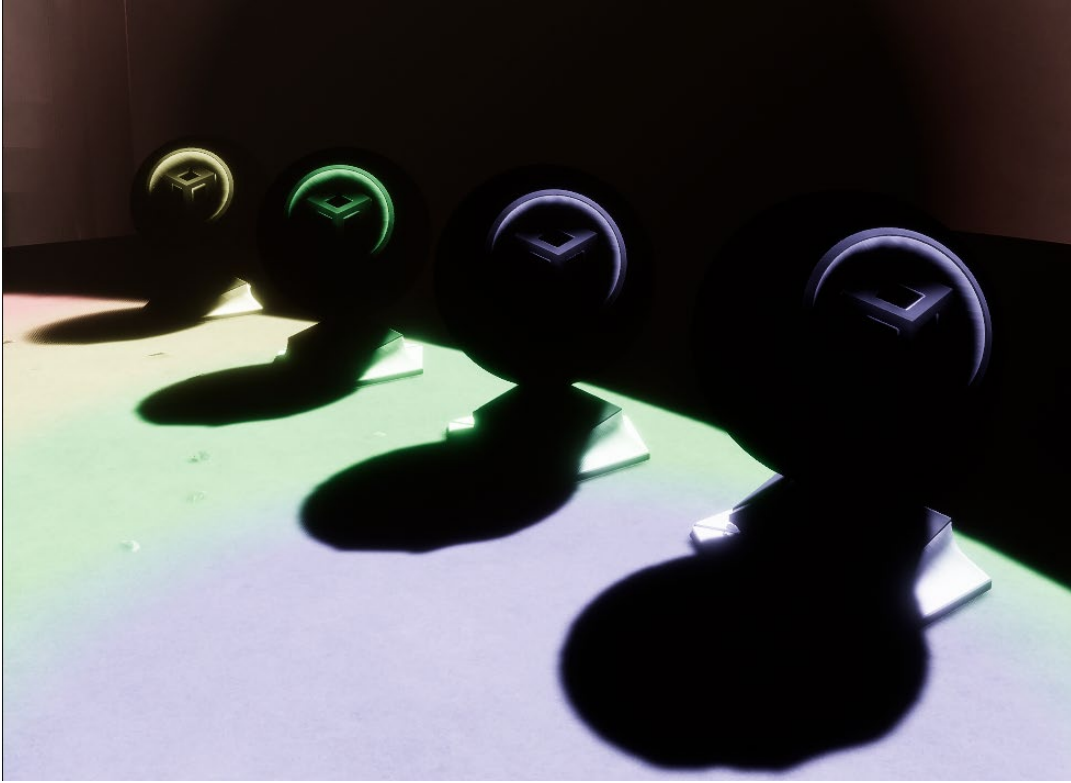
그림자 캐스케이드로 원근 앨리어싱을 줄입니다.

HDRP에서는 **Shadows** 오버라이드를 사용해 그림자 캐스케이드를 추가로 제어할 수 있습니다. 볼륨별로 캐스케이드 설정을 사용하면 각 캐스케이드가 시작하고 끝나는 지점을 조정할 수 있습니다.



Shadows 오버라이드로 그림자 캐스케이드를 조정할 수 있습니다.

Show Cascades 버튼을 토글하면 캐스케이드 분할을 더 쉽게 볼 수 있습니다. 약간의 미세 조정을 거치면 원근 앨리어싱을 최소화할 수 있습니다.



Show Cascades 옵션을 클릭해 캐스케이드 분할을 시각화한 모습입니다.

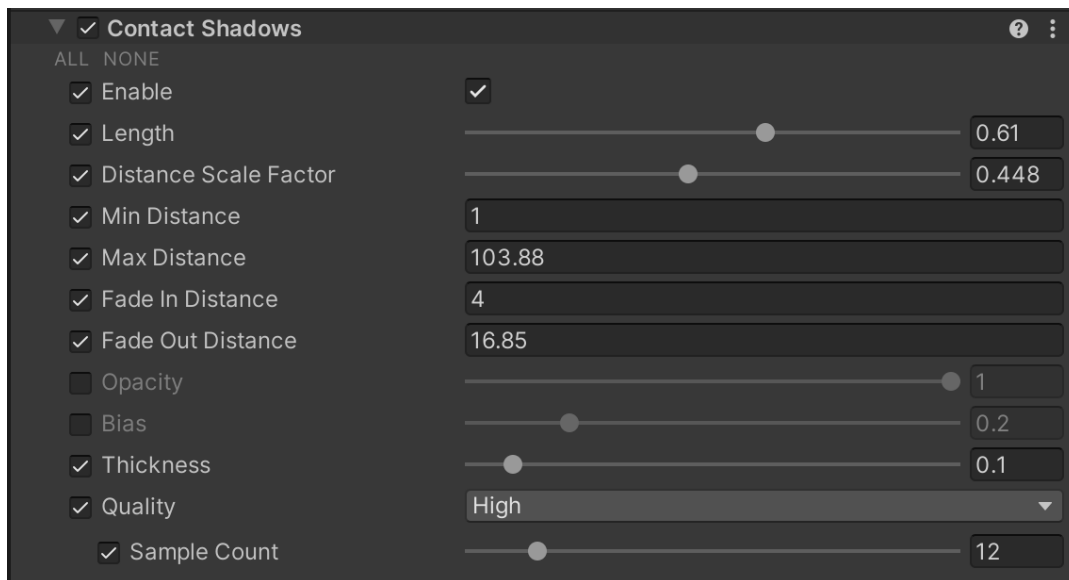
컨택트 새도우

새도우 맵은 작은 디테일을 캡처하지 못하는 경우가 있는데, 두 메시 표면이 이어지는 부분이 식별되는 에지에서 특히 그렇습니다. HDRP는 **Contact Shadows** 오버라이드를 사용하여 컨택트 새도우를 생성할 수 있습니다.

컨택트 새도우는 스크린 공간 효과로, 프레임 내 정보를 활용해 계산합니다. 프레임 외부의 오브젝트는 컨택트 새도우에 영향을 주지 않습니다. 화면에서 차지하는 공간이 작은 그림자 디테일에 컨택트 새도우를 사용하면 좋습니다.



Frame Settings에서 **Contact Shadows**를 활성화하세요. 파이프라인 에셋에서 **Lighting Quality Settings** 아래에 있는 **Sample Count** 값을 조정할 수도 있습니다.



Contact Shadows 오버라이드



컨택트 섀도우

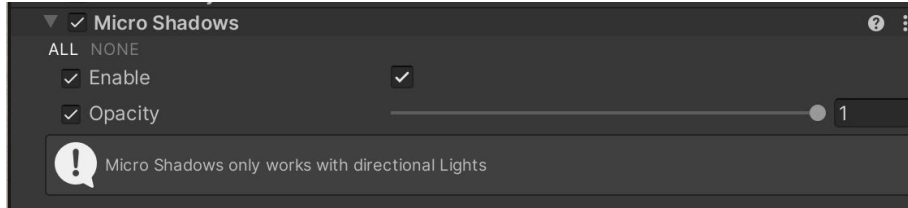


그림자 두께, 품질, 페이드 등의 설정을 조정합니다.

이 기능은 터레인 및 SpeedTree와 더 원활하게 작동하도록 개선되었습니다. [블로그](#)에서 더 자세한 내용을 알아보세요.

마이크로 새도우

HDRP는 더 작은 그림자 디테일을 머티리얼로 확장할 수 있습니다. **Micro Shadows** 오버라이드는 노멀 맵과 앰비언트 오클루전 맵을 사용하여 메시 지오메트리를 사용하지 않고도 매우 미세한 표면 그림자를 렌더링합니다.



Micro Shadows 오버라이드

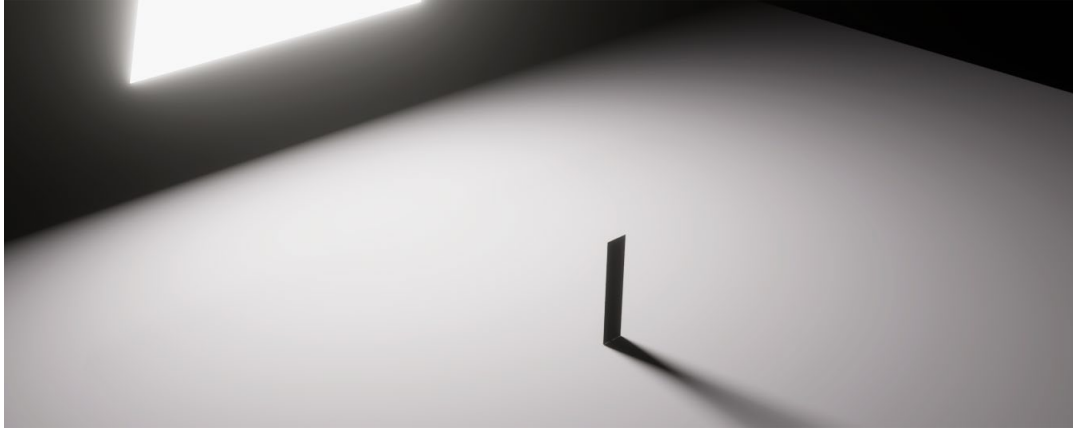
Micro Shadows 오버라이드를 씬의 볼륨에 추가하고 Opacity 값만 조정하면 됩니다. 마이크로 새도우는 방향 광원에만 연동됩니다.



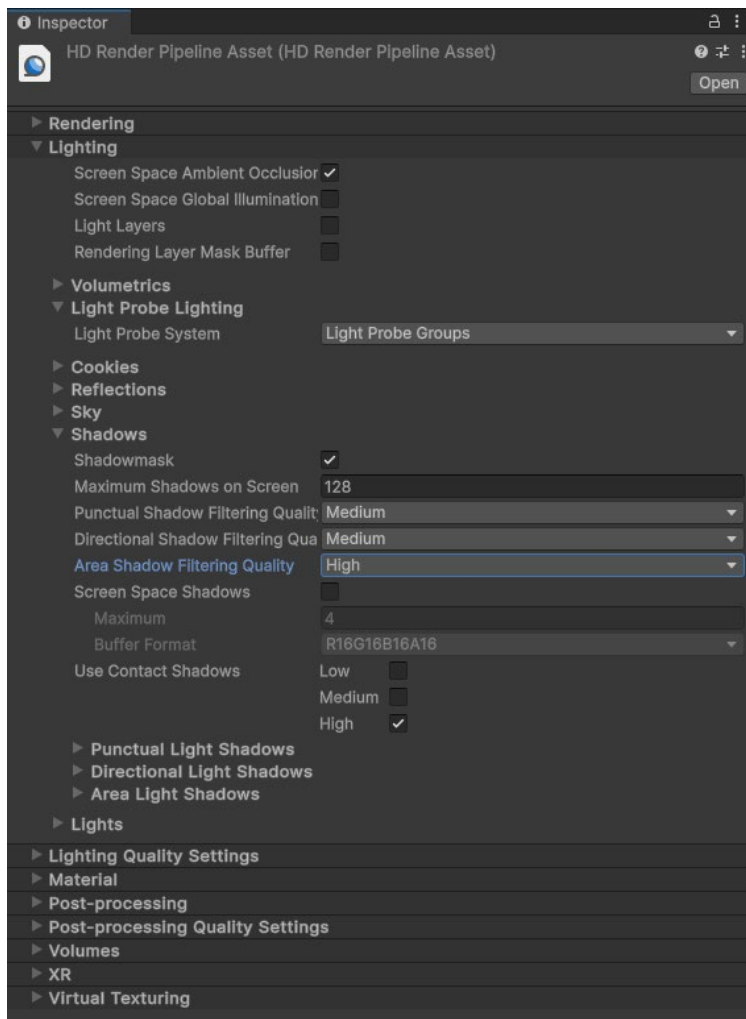
마이크로 새도우로 나뭇잎에 콘트라스트를 더합니다.

면 광원 소프트 색도우

HDRP 에셋에서 **Area Shadow Filtering Quality**를 High로 설정하면 향상된 면 광원 소프트 색도우를 사용할 수 있습니다.



면 광원으로 소프트 색도우를 향상합니다.



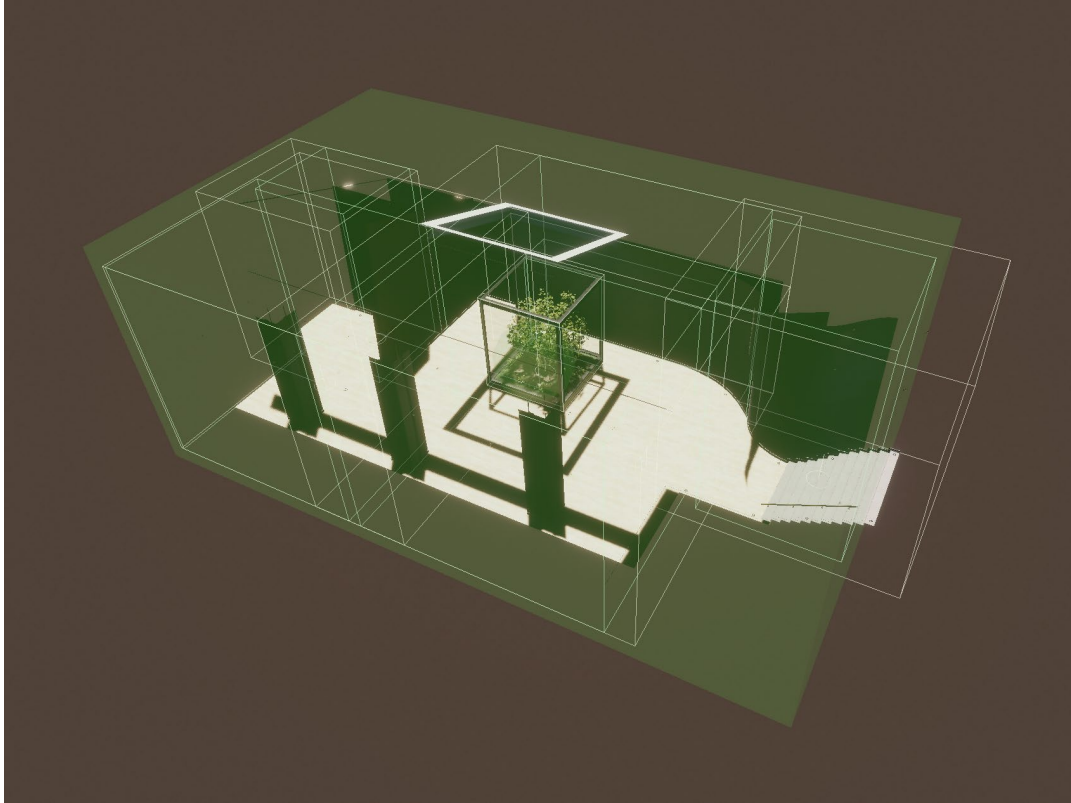
Area Shadow Filtering Quality

반사

반사를 사용하면 게임 오브젝트를 주변 환경과 통합할 수 있습니다. 일반적으로는 반사가 매끄럽고 반짝이는 표면과 관련이 있다고 생각하지만, PBR 워크플로에서의 거친 머티리얼 역시 적절하게 반사를 받아야 합니다. HDRP에서 반사를 연출하는 기법은 다음과 같습니다.

- 스크린 공간 반사
- 반사 프로브
- 하늘 반사

각 반사 유형은 리소스를 많이 사용할 수 있으므로, 용도에 따라 가장 알맞은 방법을 선택해야 합니다. 한 픽셀에 여러 반사 기법을 적용하는 경우, HDRP는 각 반사 유형의 영향을 블렌딩합니다. 인플루언스 볼륨 (Influence Volume)이라고 하는 바운딩 표면은 3D 공간을 분할하여 반사를 받는 오브젝트를 정합니다.

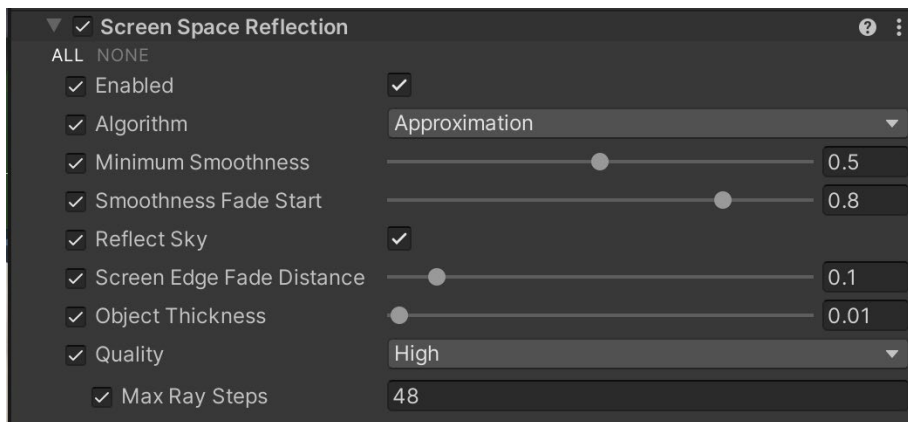


인플루언스 볼륨은 반사 프로브가 반사를 생성하는 위치를 결정합니다.

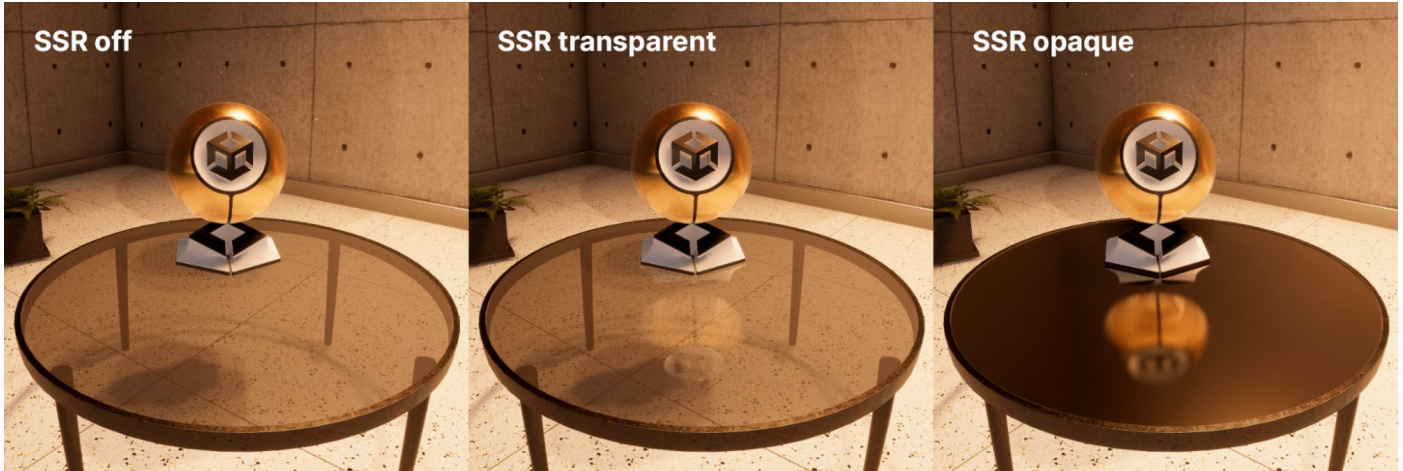
스크린 공간 반사

SSR(스크린 공간 반사)은 뎁스 버퍼와 색상 버퍼를 사용하여 반사를 계산합니다. 따라서 현재 카메라 뷰에 있는 오브젝트만 반사할 수 있으며, 화면의 특정 위치에서는 제대로 렌더링되지 않을 수 있습니다. 광택이 있는 바닥과 젖은 평면 표면은 스크린 공간 반사를 받기 좋습니다.

스크린 공간 반사는 프레임 밖의 모든 오브젝트를 무시하는데, 이로 인해 효과가 제한될 수 있습니다.



Screen Space Reflection 오버라이드



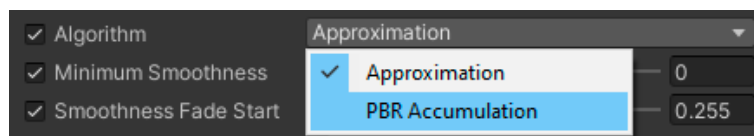
투명 재료와 불투명 재료에 스크린 공간 반사가 적용된 모습입니다.

Frame Settings(HDRP Default Settings 또는 카메라의 Custom Frame Settings)에서 **Lighting** 아래에 있는 **Screen Space Reflection** 설정을 활성화합니다. 그런 다음 **Screen Space Reflection** 오버라이드를 볼륨 오브젝트에 추가합니다.

SSR을 표시하려면 재료 표면이 **Minimum Smoothness** 값을 초과해야 합니다. 거친 재료에 스크린 공간 반사를 나타내기 위해 이 값을 낮출 수 있지만, Minimum Smoothness의 임계값을 낮추면 계산에 리소스를 더 많이 사용할 수 있습니다. Screen Space Reflection 오버라이드가 픽셀에 영향을 줄 수 없으면, HDRP는 반사 프로브를 사용하도록 폴백됩니다.

Quality 드롭다운을 사용하면 **Max Ray Steps**의 프리셋 값을 선택할 수 있습니다. Max Ray Steps 값이 높을수록 품질은 향상되지만 리소스 사용량은 늘어납니다. 다른 효과와 마찬가지로 성능과 품질의 균형을 적절히 조절하세요.

참고: 누적을 사용하는 물리 기반 알고리즘과 상대적으로 정확도가 낮은 모방 알고리즘(기본값) 중 하나를 선택할 수 있습니다.



SSR을 표시하려면 재료 표면이 Minimum Smoothness 값을 초과해야 합니다.

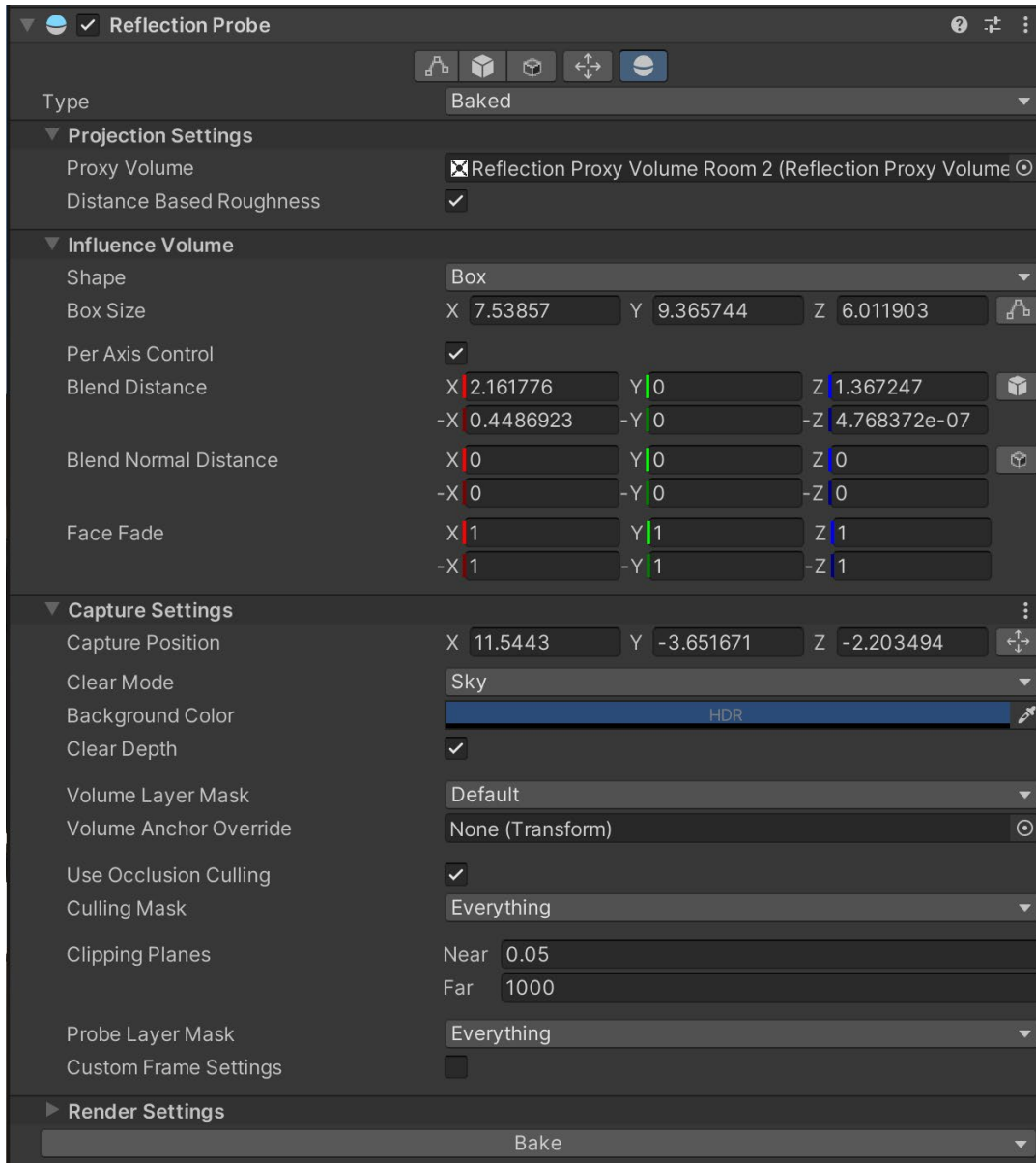
반사 프로브

반사 프로브는 이미지를 기반으로 반사를 생성하는 기법입니다. 프로브는 주변 모든 방향을 구형 뷰로 캡처하고 그 결과를 큐브맵 텍스처에 저장합니다. 셰이더는 해당 큐브맵을 사용하여 반사를 추정해 표현합니다.

각 씬에는 여러 프로브를 배치해 결과물을 블렌딩할 수 있습니다. 그러면 카메라가 환경 속에서 이동함에 따라 위치를 기반으로 반사가 변하게 됩니다.

각 프로브의 **Type**을 **Baked**나 **Real-time**으로 설정합니다.

- 베이킹된 프로브는 정적 환경에서 큐브맵 텍스처를 한 번만 처리합니다.
- 실시간 프로브는 에디터가 아닌 플레이어 런타임에 큐브맵을 생성합니다. 따라서 반사는 정적 오브젝트에 국한되지 않지만, 실시간 업데이트로 인해 리소스를 많이 사용할 수 있습니다.



Reflection Probe 컴포넌트

최적화 팁

실시간 반사 프로브를 최적화하려면 반사의 비주얼 품질에 큰 영향을 주지 않는 렌더링 기능을 비활성화합니다. 카메라 설정을 전역적으로 또는 프로브별로 **오버라이드**하면 됩니다. 업데이트 시간을 분할하는 스크립트를 만들 수도 있습니다.

Time Slicing 프로퍼티는 Unity 2022에서 Reflection Probe 컴포넌트에 추가되었습니다. 이 프로퍼티를 보려면 반사 프로브의 **Type**을 **Realtime**으로 설정하세요. 이 기능은 큐브맵 전체를 한 번에 업데이트하는 대신 프레임당 한 면만 업데이트하여 처리 부하를 분산시키고 효율성을 향상합니다. 이를 통해 성능 비용을 줄일 수 있습니다.

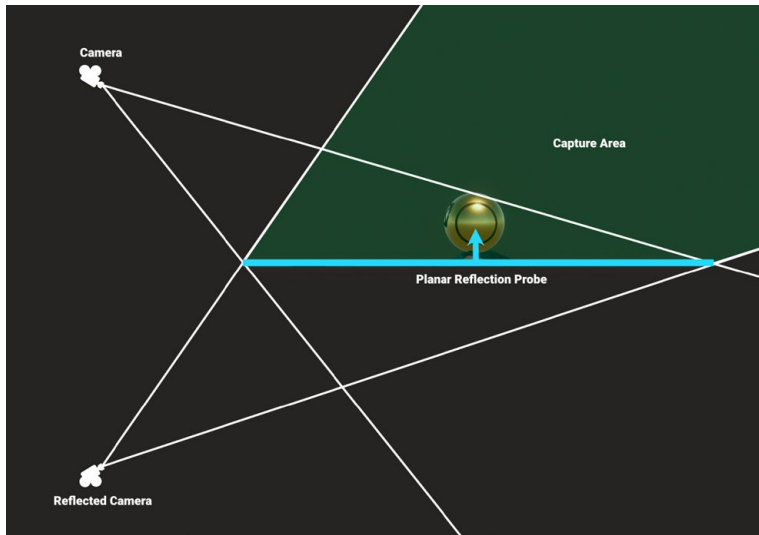
HDRP는 팔면체 투영 시 큐브 반사 프로브 캐시 배열을 **2D 텍스처 아틀라스 캐시**로 대체합니다. 이를 통해 각 반사 프로브의 해상도를 제어하여 메모리를 절약할 수 있습니다.

Influence Volume에서는 게임 오브젝트가 반사를 받을 3D 경계를 정할 수 있으며, **Capture Settings**에서는 반사 프로브가 큐브맵의 스냅샷을 생성하는 방식을 커스터마이징할 수 있습니다.

Planar Reflection Probe 컴포넌트

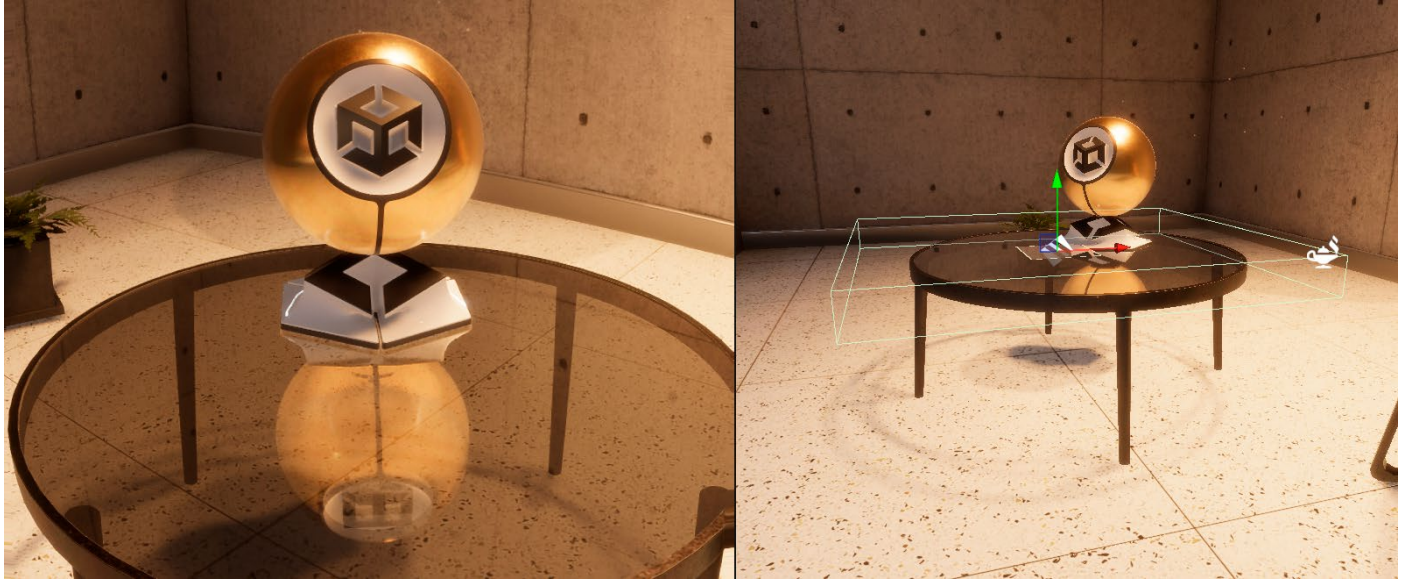
Planar Reflection Probe 컴포넌트를 사용하면 표면의 평활도를 고려하여 평평한 반사 표면을 재현할 수 있습니다. 반짝이는 거울이나 바닥의 반사를 표현하는 데 매우 좋습니다.

평면 반사 프로브는 기본 반사 프로브와 많은 부분에서 비슷하지만, 두 프로브가 작동하는 방식은 약간 다릅니다. 평면 반사 프로브는 환경을 큐브맵으로 캡처하는 대신 프로브의 반사면을 통해 반사된 카메라의 뷰를 재현합니다.



그런 다음 결과로 얻은 좌우 대칭된 이미지를 2D Render Texture에 저장하고, 이를 직사각형 프로브의 경계에 드로우하여 평면 반사를 만듭니다.

평면 반사 프로브는 평면을 통해 카메라를 반사하여 좌우 대칭된 이미지를 캡처합니다.



평면 반사 프로브는 평평한 오브젝트의 반사를 구현합니다.

하늘 반사

오브젝트가 주변에 있는 반사 프로브의 영향을 받지 않는 경우, 하늘 반사로 폴백됩니다.



반사 프로브는 주변 공간을 보여 주는 반면 하늘 반사는 Gradient Sky를 반사합니다.

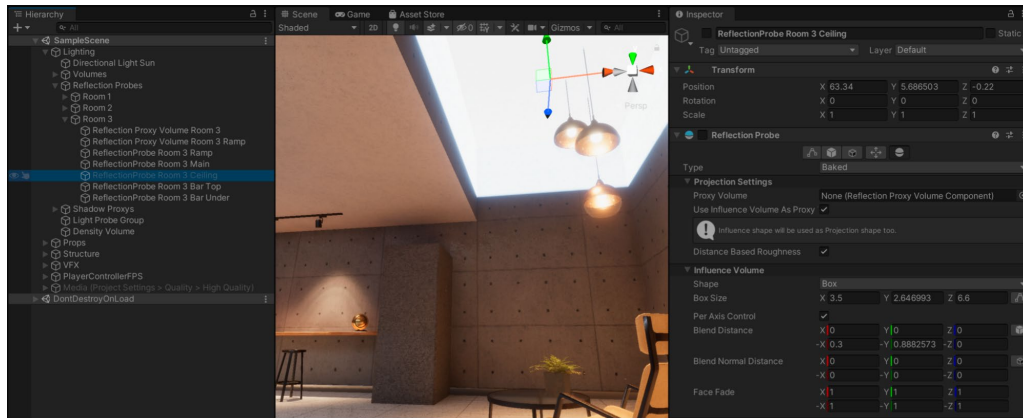
반사 계층 구조

가장 높은 품질의 반사를 만들기 위해 HDRP는 각 픽셀을 가장 정확하게 표현하는 기법을 사용하며, 이를 다른 기법과 혼합해 사용할 수 있습니다. HDRP에서는 가중치 우선순위를 바탕으로 세 가지 반사 방법 (스크린 공간 반사, 반사 프로브, 하늘 반사)을 확인합니다. 이렇게 반사를 확인하는 시퀀스를 **반사 계층 구조**라고 합니다.

한 기법으로 픽셀의 반사를 완전히 정하지 못하는 경우, HDRP는 다음 기술로 폴백합니다. 다시 말해 스크린 공간 반사는 반사 프로브로 폴백되며, 이후 하늘 반사로 폴백됩니다.

반사 프로브에 인플루언스 볼륨을 올바르게 설정해야 합니다. 그렇지 않으면 하늘 반사가 의도한 바와 다르게 적용되어 인해 빛 번짐 현상이 발생할 수 있습니다.

샘플 씬의 세 번째 공간에서 이를 뚜렷이 확인할 수 있습니다. 반사 프로브 중 하나를 비활성화하거나 인플루언스 볼륨을 바꾸면 반사가 강제로 하늘 반사로 폴백됩니다. 그러면 밝은 HDRI 하늘로 인해 강한 반사 효과가 발생해 반사광이 씬을 뒤덮습니다.



세 번째 공간 천장에서 반사 프로브를 비활성화하면 의도치 않은 빛 번짐 현상이 발생합니다.

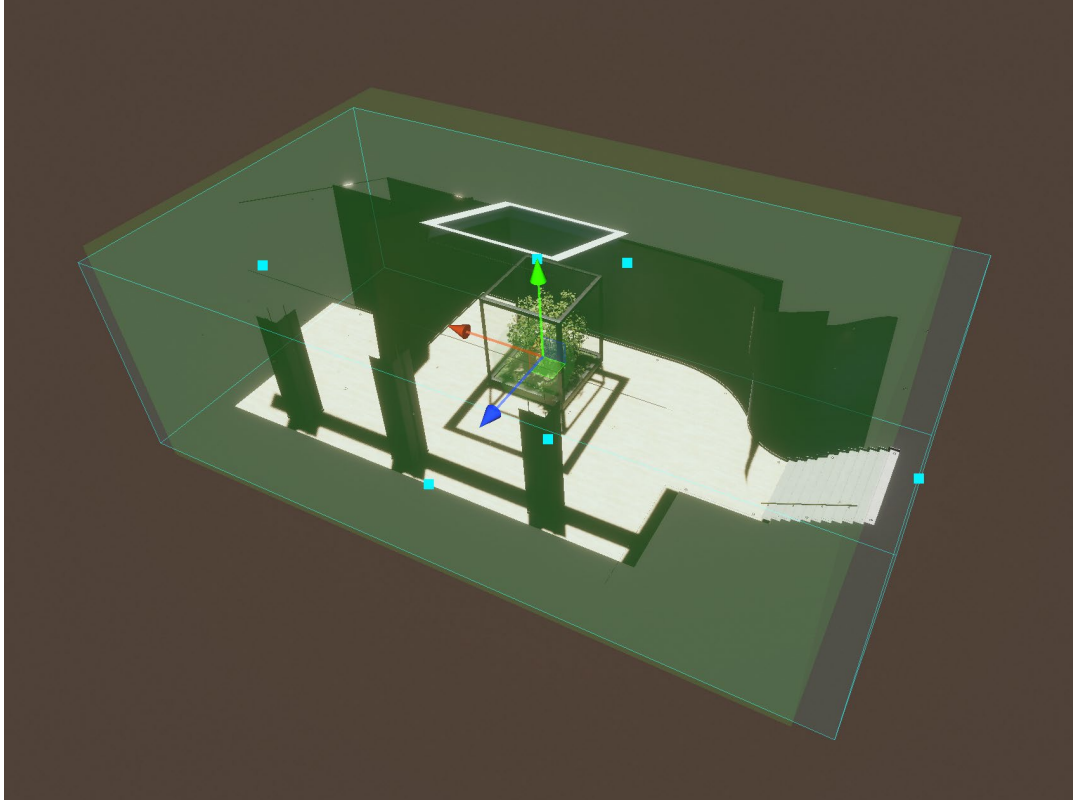
반사 계층 구조 결정에 관한 자세한 내용은 [HDRP에서의 반사](#) 기술 자료 페이지를 참조하시기 바랍니다.



프록시 볼륨

반사 프로브의 캡처 포인트는 고정되어 있고 반사 프로브 근처의 카메라 위치와 일치하는 경우가 거의 없기 때문에, 결과로 나타난 반사에 눈에 띄는 원근 변화가 있을 수 있습니다. 결과적으로 반사가 환경과 관련이 없는 것처럼 보일 수 있습니다.

프록시 볼륨을 사용하면 이 문제를 부분적으로 해결할 수 있습니다. 프록시 볼륨은 카메라의 위치를 기반으로 프록시 볼륨 내에서 반사를 더 정확하게 재투영합니다.



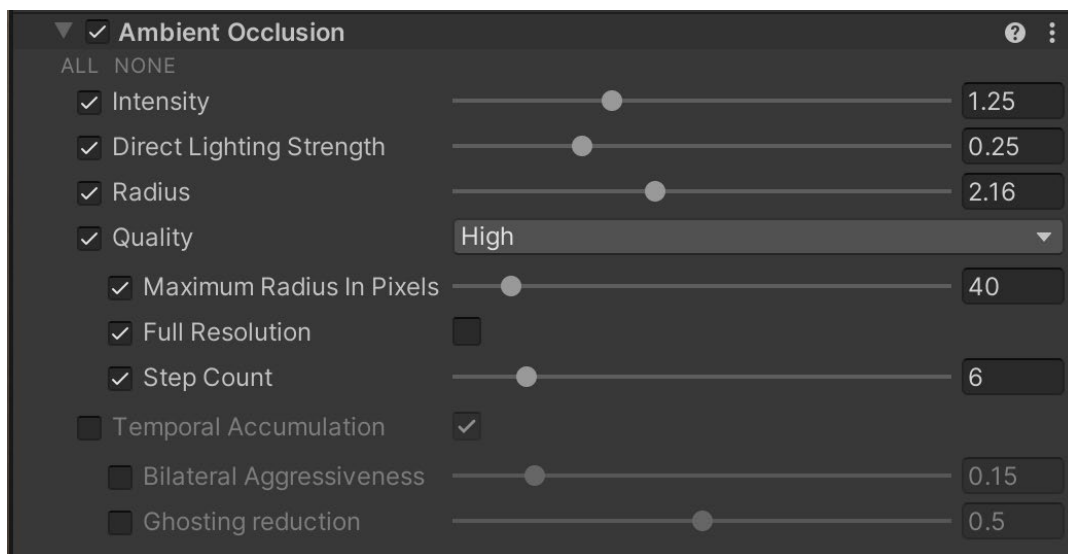
프록시 볼륨은 공간의 월드 공간 위치와 일치하도록 큐브맵을 재투영합니다.

실시간 조명 효과

HDRP는 또한 볼륨 시스템을 통해 사용할 수 있는 몇 가지 실시간 조명 효과를 제공합니다. 로컬 볼륨이나 글로벌 볼륨을 선택한 다음 **Add Override > Lighting**에서 원하는 효과를 추가하면 됩니다.

스크린 공간 앰비언트 오클루전

앰비언트 오클루전은 서로 가까운 주름, 구멍, 표면에서 발생하는 어두워지는 현상을 시뮬레이션합니다. 주변광을 차단하는 영역은 가려져서 보입니다.



Ambient Occlusion 오버라이드



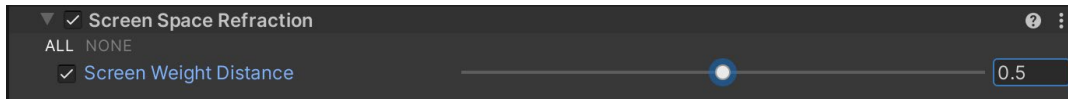
스폰자 아트리움의 스크린 공간 앰비언트 오클루전 시각화

Unity의 라이트매퍼를 사용해 정적 지오메트리에 대한 앰비언트 오클루전을 베이킹할 수 있지만, HDRP는 실시간으로 작동하는 **Screen Space Ambient Occlusion 오버라이드**를 추가로 제공합니다. 스크린 공간 앰비언트 오클루전은 스크린 공간 효과이기 때문에, 프레임 내의 정보만을 바탕으로 효과가 만들어집니다. SSAO는 카메라 시야각 밖의 모든 오브젝트를 무시합니다.

Frame Settings에 있는 **Lighting**에서 **Screen Space Ambient Occlusion**을 활성화한 후, 로컬 볼륨이나 글로벌 볼륨에서 **Add Override**를 클릭하고 **Lighting > Ambient Occlusion**을 선택하면 됩니다.

스크린 공간 굴절

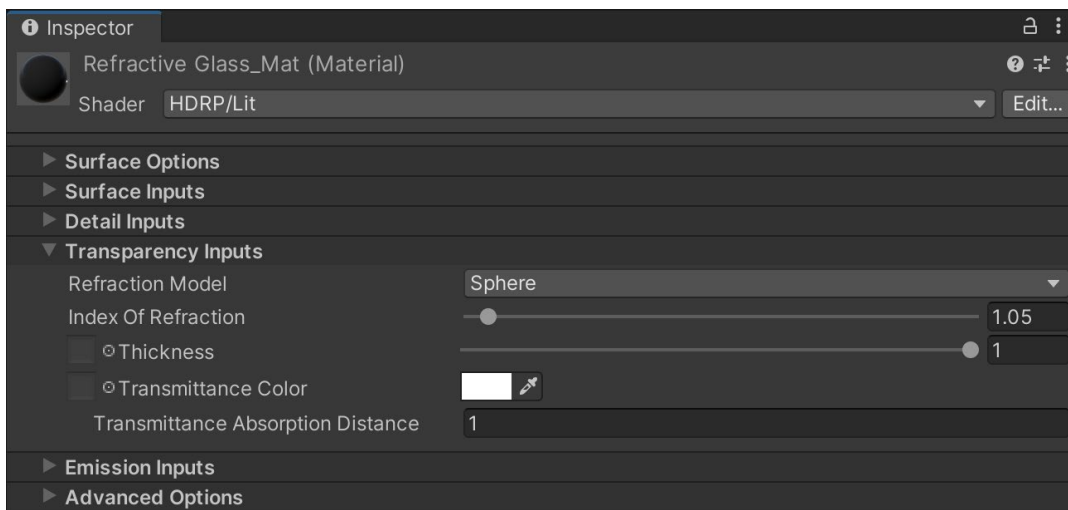
Screen Space Refraction 오버라이드는 빛이 공기보다 밀도가 높은 매체를 통과할 때 어떤 현상이 일어나는지 시뮬레이션하는 데 사용할 수 있습니다. HDRP의 스크린 공간 굴절은 덤스 버퍼와 색상 버퍼를 사용하여 유리나 같은 투명한 머티리얼을 통과하는 굴절을 계산합니다.



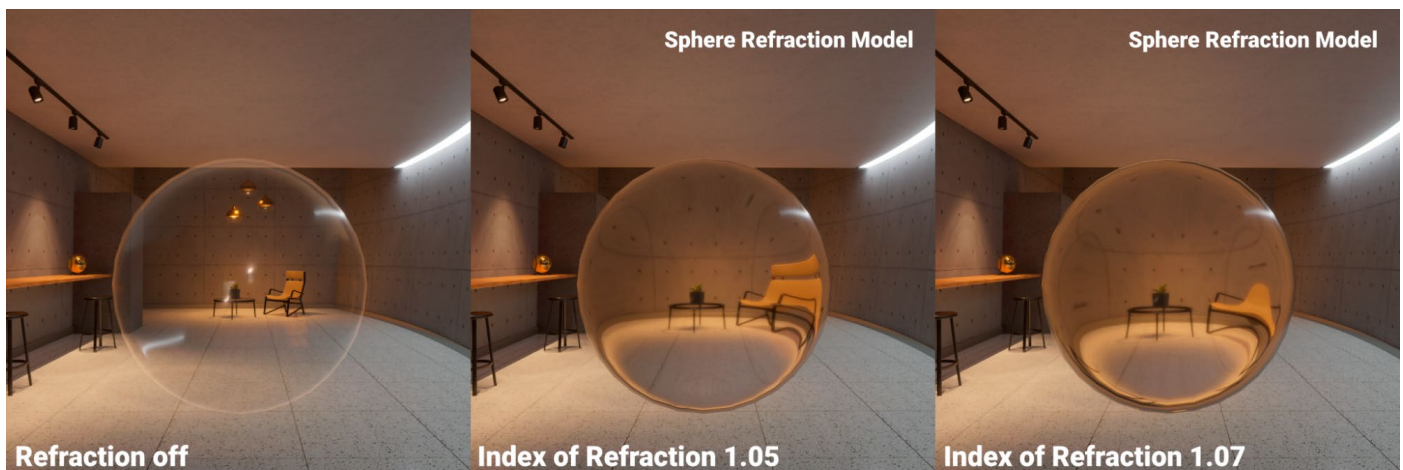
Screen Space Refraction 오버라이드

HDRP/Lit 셰이더를 사용해 이 효과를 활성화하려면, 머티리얼의 Surface Type이 **Transparent**여야 합니다.

그런 다음 **Transparency Inputs**에서 **Refraction Model**과 **Index Of Refraction**을 선택합니다. 속이 꽉 찬 오브젝트의 경우, Refraction Model에서 **Sphere**를 선택합니다. 속이 빈 오브젝트의 경우 **Thin**(거품 등)이나 **Box**(약간 두께가 있는 경우)를 선택합니다.



굴절을 제어하는 Transparency Inputs



스크린 공간 굴절

광원 렌더링 레이어

HDRP의 렌더링 레이어를 사용하여 어느 광원, 데칼, 효과가 씬의 특정 렌더러에 영향을 주는지 제어할 수 있습니다. 렌더링 레이어는 특정 렌더러에만 영향을 주도록 광원 또는 효과를 제한하는 [LayersMask](#)입니다.

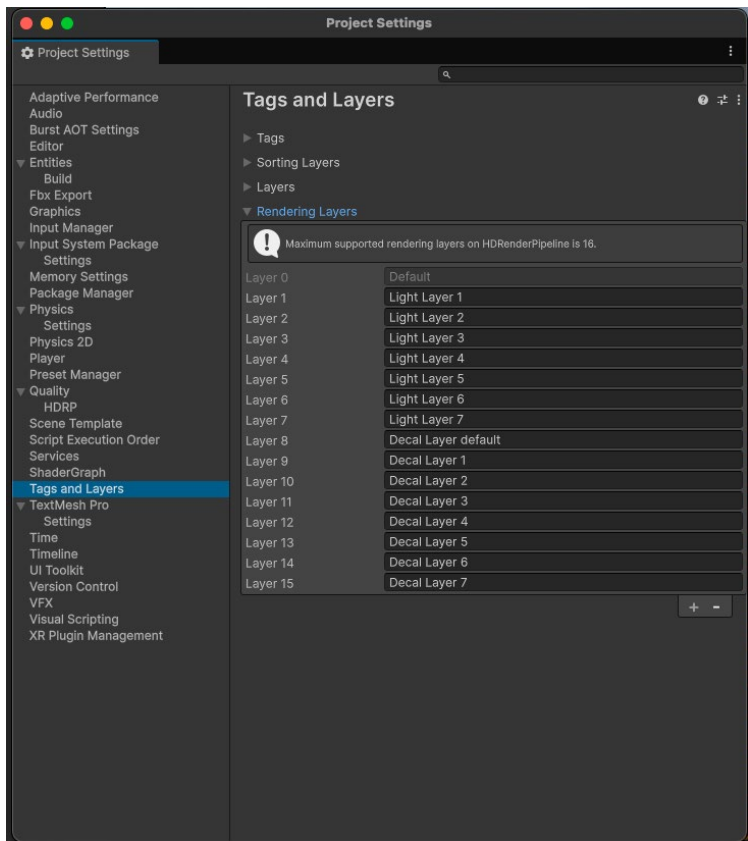


렌더링 레이어 16개와 32개 비교

모든 렌더러(메시 또는 오브젝트)는 최대 32개의 렌더링 레이어에 속할 수 있으나, HDRP 광원과 효과는 처음 16개의 레이어만 지원합니다. 스크립팅 또는 구성을 위해 32개의 레이어 중 어느 레이어에나 렌더러를 할당할 수 있지만, 처음 16개의 레이어만 조명, 데칼 또는 기타 HDRP 효과에 영향을 줍니다.

렌더링 레이어 설정

렌더링 레이어를 만들고 이름을 지정하려면 **Edit > Project Settings > Tags and Layers**로 이동합니다. **Rendering Layers**를 열고 간편한 관리를 위해 레이어에 커스텀 이름을 할당합니다.



Rendering Layers 기능을 활성화합니다.

광원에 렌더링 레이어를 활성화하려면 **Edit > Project Settings > Graphics > Pipeline Specific Settings > HDRP**로 이동합니다. **Frame Settings (Default Values)**의 **Lighting** 아래에서 **Light Layers**를 활성화합니다.

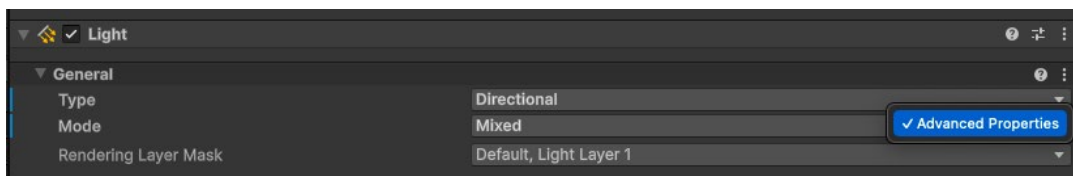
그런 다음 **HDRP Asset**에서 **Light Layers**를 활성화합니다.

렌더링 레이어를 데칼에 사용하려면 **Decal Layers** 기술 자료를 참조하세요.

광원에 렌더링 레이어 사용

Light Layers를 활성화한 후에는 광원과 메시 렌더러에 할당할 수 있습니다.

인스펙터에서 원하는 광원을 선택한 다음 인스펙터의 **General** 섹션에서 옵션 메뉴(:)의 **Advanced Properties**를 활성화합니다.



Advanced Properties에서 렌더링 레이어를 표시합니다.

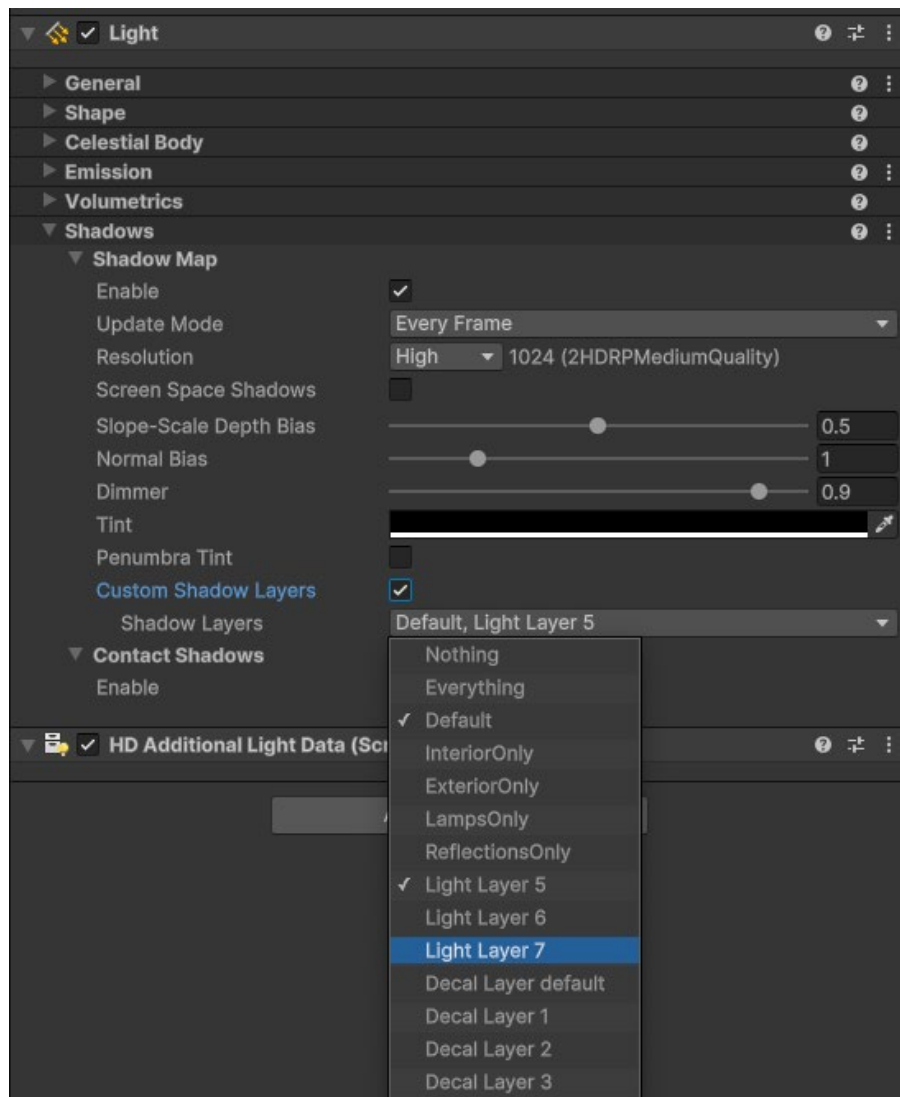
Rendering Layer Mask 드롭다운을 사용하여 해당 광원의 영향을 받는 **Light Layer**를 지정합니다.

메시 렌더러 또는 터레인을 할당하려면 해당 **인스펙터**의 **Rendering Layer Mask** 드롭다운을 사용합니다. 광원은 양쪽이 동일한 렌더링 레이어를 사용하는 경우에만 메시 렌더러 또는 터레인에 영향을 줍니다.

그림자용 렌더링 레이어 사용

HDRP는 기본적으로 광원의 렌더링 레이어를 그림자 렌더링에 연결합니다. 따라서 광원이 빛을 비추는 모든 것에는 해당 레이어를 기반으로 그림자가 드리워지게 됩니다.

그림자를 광원에서 분리해야 하는 경우, Light의 **Shadows** 설정에서 **Custom Shadow Layers** 설정을 활성화할 수 있습니다. 이 기능을 사용하면 그림자 전용 렌더링 레이어를 할당할 수 있습니다.



렌더링 레이어를 사용하여 광원 효과를 분리합니다.

이 예시에서는 실제로 그림자 렌더링을 사용하는 예시를 확인할 수 있습니다. 왼쪽에서는 광원 쿠키가 전구 위에 의도치 않은 셀프 색도잉을 유발합니다. 가운데에서는 전구의 메시 렌더러를 별도의 렌더링 레이어에 할당하여 광원 쿠키가 더 이상 전구에 영향을 주지 않습니다. 그런 다음 오른쪽에서는 전구와 동일한 렌더링 레이어에 별도의 점 광원이 추가되어 조명을 의도한 대로 복원합니다.



렌더링 레이어를 사용하여 조명과 그림자를 커스터마이징합니다.

자세한 내용은 [Unity의 고품질 조명 비품](#)에 관한 전문가 가이드를 참조하세요.

레이트레이싱 및 패스트레이싱

레이트레이싱은 기존의 래스터화보다 더 현실감 있는 렌더링을 구현하는 기술입니다. 기존에는 레이트레이싱을 계산하는 데 리소스가 많이 소모됐지만, 최근 하드웨어 개발이 가속화되면서 실시간 애플리케이션에서도 레이트레이싱을 사용할 수 있게 되었습니다.

HDRP는 레이트레이싱과 래스터화 렌더링이 결합된 하이브리드 레이트레이싱 시스템을 사용합니다. DirectX 12를 지원하며, 호환되는 NVIDIA RTX™ 및 AMD Radeon™ PRO GPU가 필요합니다.

구체적인 시스템 요구 사항 목록은 [레이트레이싱 시작하기](#)를 참조하시기 바랍니다.

Unity 6의 새로운 기능

- **정식 제작에 사용 가능한 레이트레이싱:** HDRP는 DirectX 12 및 특정 콘솔 플랫폼을 위한 레이트레이싱을 지원합니다. 자세한 내용은 해당 콘솔의 기술 자료를 참조하세요.
- **향상된 레이트레이싱 효과:** 레이트레이싱을 거친 앰비언트 오클루전, 전역 조명, 반사 및 그림자 개선으로 더 정확하고 사실적인 조명과 셰이딩을 제공합니다.
- **입체각 컬링:** 이제 입체각 컬링을 활성화하여 가속 구조에서 작고 멀리 있는 인스턴스를 폐기할 수 있습니다. 이는 [HDRP Ray Tracing Volume 설정](#)의 설정과 [EnableSolidAngleCulling](#)을 통한 스크립팅 API를 통해 활성화됩니다. 입체각 컬링을 활성화하면 손쉽게 레이트레이싱 성능을 개선할 수 있습니다.
- **인라인 레이트레이싱 및 셰이더 쿼리:** 이제 인라인 레이트레이싱 및 셰이더를 통해 하드웨어 가속 광선 쿼리를 활용할 수 있습니다. 이에 따라 커스텀 레이트레이싱 그림자와 같은 커스텀 효과를 만들 수 있는 새로운 가능성이 열립니다.

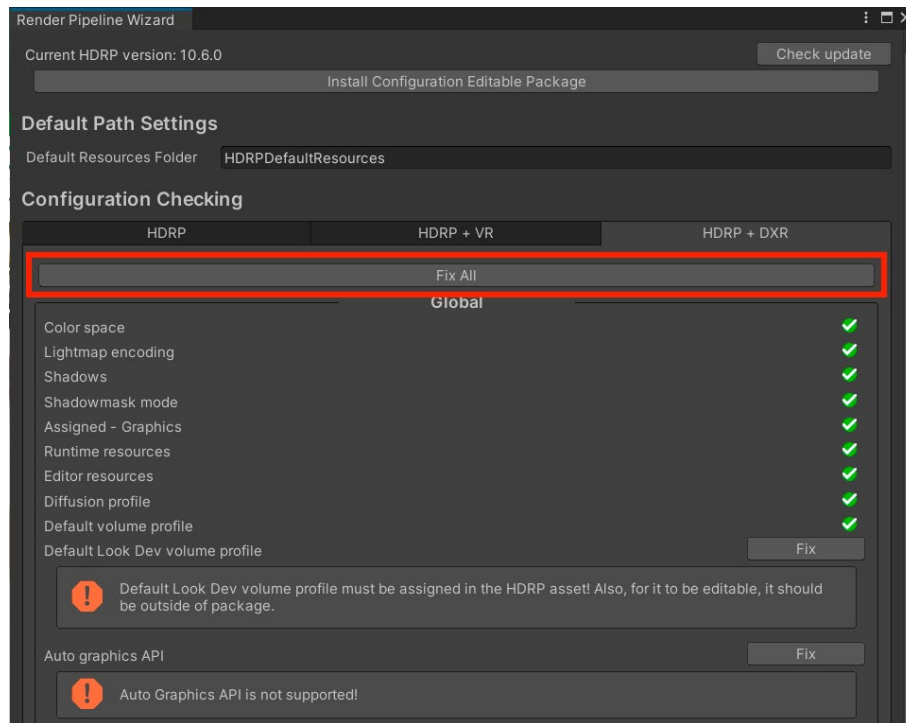
- **데칼 지원:** 이제 데칼이 레이트레이싱 반사와 전역 조명에 완전히 통합되었습니다. 예를 들어, 큰 노란색 데칼은 이제 가까운 표면을 따뜻한 느낌의 반사광으로 물들여 더 사실적인 렌더링을 제공합니다.

이제 데칼 프로젝터를 HDRP 패스트레이서와 함께 사용하여, 기반 표면을 수정하지 않고도 물 웅덩이, 먼지, 총알 구멍 및 기타 머티리얼 효과를 추가할 수 있습니다.
- **최적화된 가속 구조:** 이제 레이트레이싱이 복잡한 씬에 보다 적은 메모리를 사용하여 실시간 성능을 향상합니다. 이는 새로운 작은 BLAS 할당자(작은 메시에 낭비되는 메모리 감소), **BLAS 압축**(정적 메시에 사용되는 메모리 감소), 새로운 **메모리 최소화 플래그**와 같은 여러 최적화의 결과입니다.
- **확장된 하드웨어 지원:** NVIDIA GeForce RTX GPU, NVIDIA RTX PRO GPU, 일부 AMD Radeon™ RX 및 PRO 시리즈 GPU, Windows PC용 DirectX 12 및 DXR 지원 GPU, 특정 콘솔 플랫폼(자세한 내용은 해당 콘솔의 기술 자료 참조)에서 완전한 레이트레이싱 하드웨어 가속을 사용할 수 있습니다.

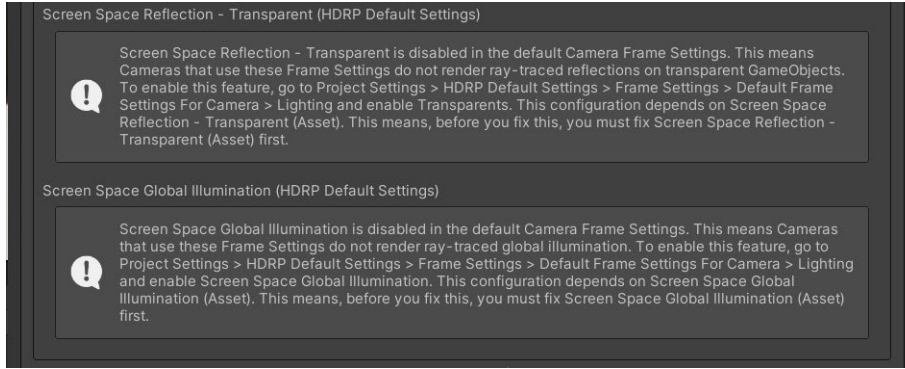
설정

레이트레이싱을 활성화하려면 HDRP 프로젝트의 기본 그래픽스 API를 DirectX 12로 변경해야 합니다.

Render Pipeline Wizard를 엽니다(**Window > Rendering > HDRP Wizard**). **HDRP + DXR** 탭에서 **Fix All**을 클릭하면 에디터를 다시 시작하라는 메시지가 표시됩니다.



렌더 파이프라인 마법사에서 레이트레이싱을 활성화합니다.



지침에 따라 비활성화된 기능을 수정합니다.

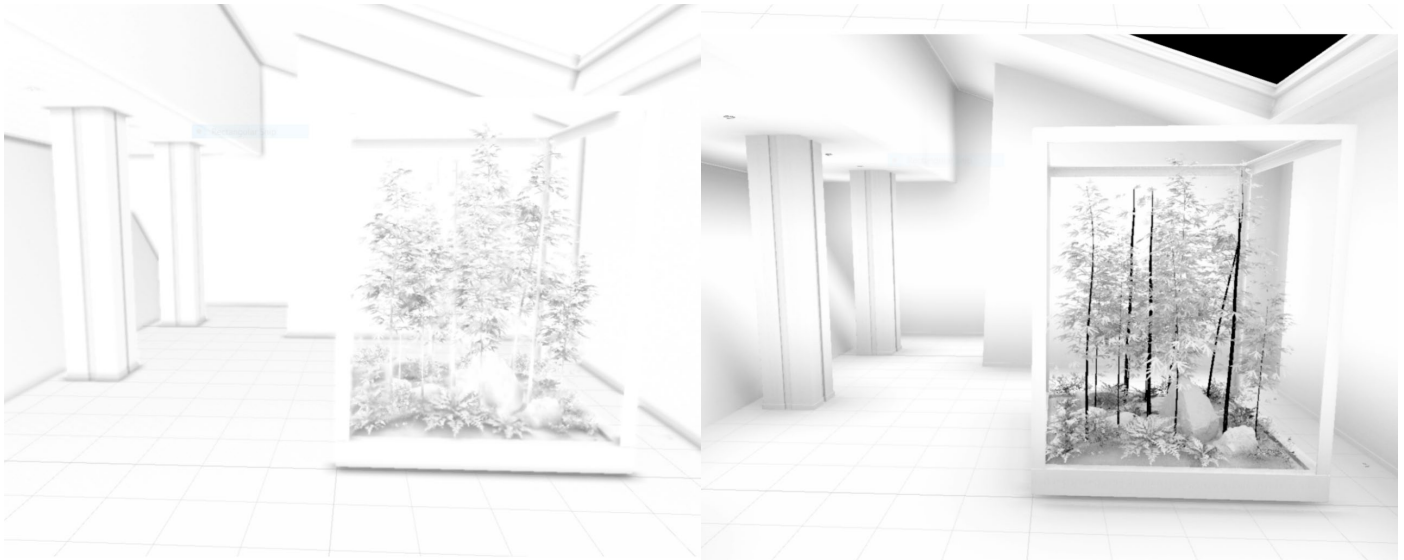
수동으로 레이트레이싱을 설정할 수도 있습니다.

프로젝트에서 레이트레이싱을 활성화했으면 **HDRP Global Settings**나 **Camera Frame Settings**에도 레이트레이싱이 활성화되어 있는지 확인합니다. **Build Settings**에서 호환되는 64비트 아키텍처를 사용하도록 설정하고, **Edit > Rendering > Check Scene Content for HDRP Ray Tracing**에서 씬 오브젝트를 확인합니다.

오버라이드

레이트레이싱을 활성화하면 몇 가지 새로운 볼륨 오버라이드가 추가되고 HDRP의 다양한 기존 오버라이드가 개선되는 효과가 있습니다.

- **레이트레이싱 앰비언트 오클루전:** 레이트레이싱 앰비언트 오클루전이 스크린 공간 앰비언트 오클루전을 대체합니다(아래의 실시간 조명 효과 참고). SSAO와 달리 레이트레이싱 앰비언트 오클루전을 사용하면 화면 밖의 지오메트리를 사용하여 오클루전을 생성할 수 있습니다. 따라서 프레임의 가장자리로 갈수록 효과가 사라지거나 부정확해지는 문제를 해결할 수 있습니다.



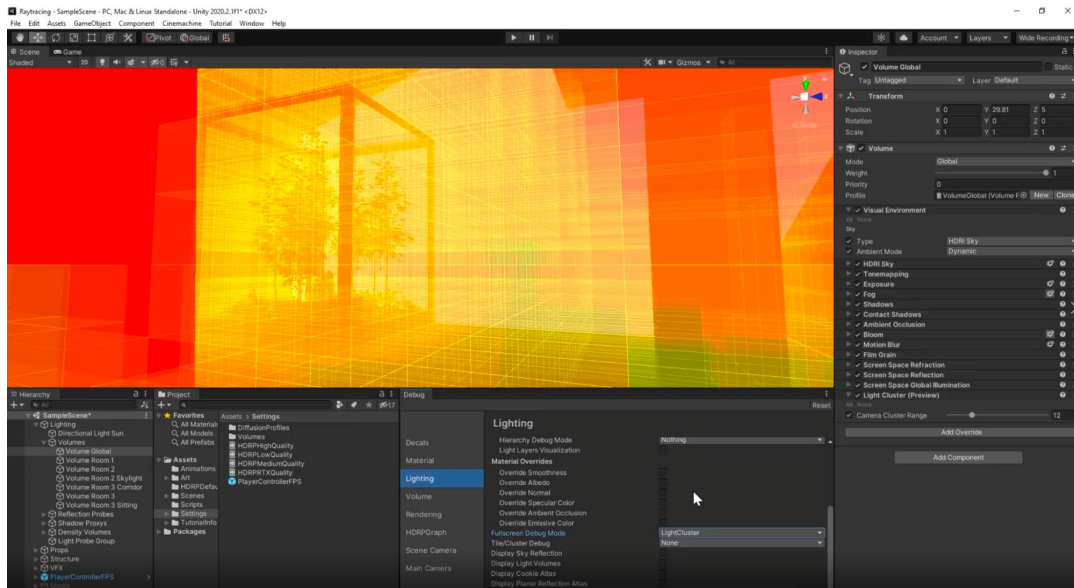
스크린 공간 앰비언트 오클루전(왼쪽 이미지)과 레이트레이싱 앰비언트 오클루전(오른쪽 이미지) 비교

- **광원 클러스터:** 레이트레이싱 반사, GI, SSS, 재귀적 렌더링을 사용할 때는 광원을 광원 클러스터에 효율적으로 저장하여 결함을 방지하고 성능을 최적화해야 합니다. HDRP는 카메라를 중심으로 하는 축 정렬 그리드로 씬을 나눕니다.

HDRP는 이 구조를 사용하여 광선이 표면에 닿을 때마다 조명에 영향을 줄 수 있는 집합적인 로컬 조명을 결정합니다. 그런 다음 특정 효과(레이트레이싱 반사, 레이트레이싱 전역 조명 등)에 대한 빛 반사를 계산할 수 있습니다.

Camera Cluster Range 볼륨 오버라이드를 사용하면 고려해야 하는 게임 오브젝트와 광원이 포함되도록 이 구조의 범위를 변경할 수 있습니다.

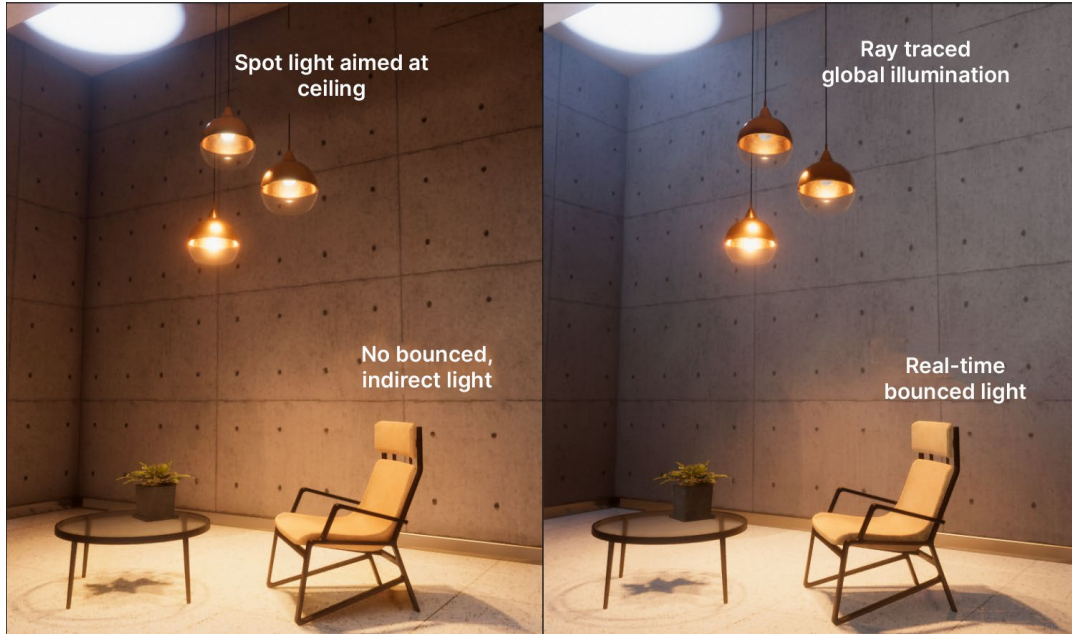
Window > Analysis > Rendering Debugger > Lighting > Full Screen Debug mode로 이동하여 **HDRP Debug** 모드를 사용할 수 있습니다. HDRP Debug 모드를 사용하면 광원 클러스터 셀을 시각화할 수 있습니다. 여기서 빨강계 표시되는 부분은 광원의 수가 HDRP 에셋에서 설정된 **Maximum Lights per Cell** 값에 도달한 부분을 나타냅니다. 해당 설정을 조정하여 의도치 않은 빛 번짐 현상이나 결함을 줄일 수 있습니다.



디버그 모드의 레이트레이싱 광원 클러스터

- **레이트레이싱 전역 조명:** 반사된 간접 조명 시뮬레이션에서 SSGI 및 라이트 프로브 대신 사용할 수 있습니다. 레이트레이싱 전역 조명은 실시간으로 계산되므로, 라이트맵을 베이킹하는 긴 오프라인 프로세스를 거치지 않고도 비슷한 결과를 얻을 수 있습니다.

여러 번의 빛 반사와 여러 개의 샘플을 활용해야 하는 복잡한 실내 환경의 경우, Quality 설정을 사용합니다. 하나의 샘플과 한 번의 반사로 제한되는 Performance 모드는 대부분의 조명이 기본 방향 광원에서 오는 외부 환경에 적합합니다.



레이트레이싱 전역 조명을 활용해 반사되는 조명을 실시간으로 확인합니다.

- **레이트레이싱 반사:** 레이트레이싱 반사를 사용하면 반사 프로브나 스크린 공간 반사를 사용하는 것보다 더 높은 품질의 반사를 구현할 수 있습니다. 화면 밖의 메시도 반사 결과에 올바르게 나타납니다.

Minimum Smoothness 값과 **Smoothness Fade Start** 값을 조정하면 매끄러운 표면이 레이트레이싱 반사를 받기 시작하는 임계값을 수정할 수 있습니다. 필요한 경우에는 **Bounces** 값을 높일 수 있지만, 성능 문제를 고려해야 합니다.

아래 이미지는 두 개의 거울이 서로를 반사하는 등의 ‘무한 거울’ 환경에서 레이트레이싱 반사의 효과를 보여 주는 예시입니다. 일련의 이미지를 통해 반사 횟수가 1, 3, 6, 8번으로 복잡해지는 과정을 볼 수 있습니다. 반사 횟수가 늘어나면 반사광이 멀어지며 희미해져 깊이감이 생깁니다.



1 Bounce



3 Bounces



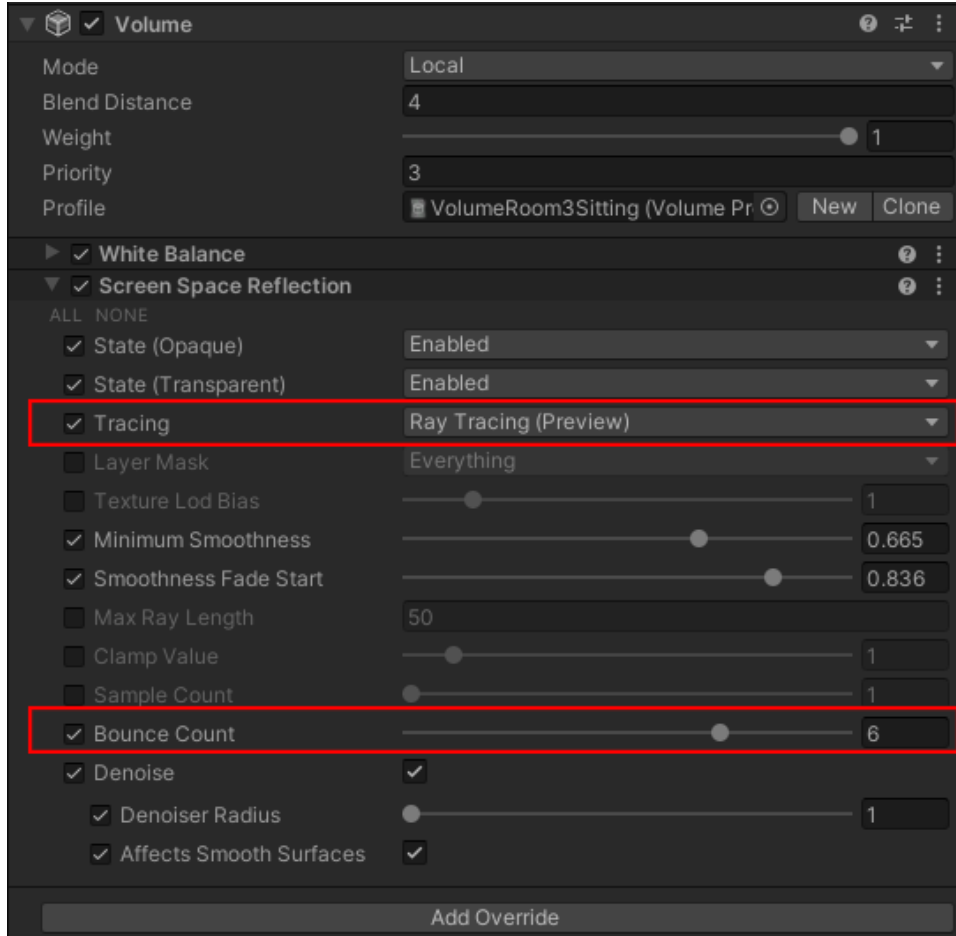
6 Bounces



8 Bounces



레이트레이싱 반사에는 화면 외부 오브젝트도 포함될 수 있습니다.



레이트레이싱으로 스크린 공간 반사를 향상할 수 있습니다.

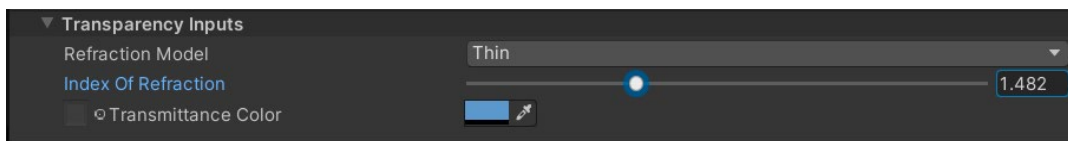
- **레이트레이싱 그림자:** 방향 광원, 점 광원, 스폿 광원, 사각 면 광원의 레이트레이싱 그림자는 모든 불투명한 게임 오브젝트의 새도우 맵을 대체할 수 있습니다. 방향 광원은 투명하거나 반투명한 게임 오브젝트에서도 레이트레이싱 그림자를 드리울 수 있습니다.

레이트레이싱은 실제처럼 캐스터와의 거리가 멀어질수록 부드럽게 표현되는 자연스러운 그림자를 만들 수 있습니다.



레이트레이싱 그림자는 캐스터에서 멀수록 부드러워져서 새도우 매핑과는 다른 효과가 나타납니다. 방향 광원, 점 광원, 스폿 광원은 반투명한 그림자를 생성할 수도 있습니다.

HDRP의 방향 광원은 색이 있는 반투명한 그림자를 생성할 수도 있습니다. 아래 이미지에서는 유리 표면으로부터 색을 띤 그림자가 바닥에 드리우는 것을 찾아볼 수 있습니다.



투명한 새도우 캐스터에 Transmittance Color를 사용합니다.



방향 광원으로 레이트레이싱 색상 그림자를 생성할 수 있습니다(오른쪽).

HDRP의 레이트레이싱 기능을 살펴보려면 [HDRP를 사용한 레이트레이싱 활성화](#)를 시청해 보세요. HDRP 마이크로사이트의 [레이트레이싱 기술 자료](#)에서도 자세한 내용을 확인할 수 있습니다.

인라인 레이트레이싱 지원(PC 및 콘솔)

[Unity 레이트레이싱 API](#)가 Windows 및 콘솔 플랫폼을 타게팅할 때 모든 셰이더 단계에 인라인 레이트레이싱 지원을 제공하도록 확대되었습니다.

레이트레이싱 파이프라인

HDRP의 각종 레이트레이싱 효과(그림자, 반사 등)는 모두 Unity 레이트레이싱 API와 기존의 ‘레이트레이싱 파이프라인’을 사용하여 구현됩니다.

레이트레이싱 파이프라인은 씬의 지오메트리를 나타내며 GPU가 광선 교차 지점을 효율적으로 찾도록 도와주는 [레이트레이싱 가속 구조](#)를 만들어서 작동합니다.

광선이 오브젝트에 닿으면 ‘히트 셰이더’가 셰이딩을 계산하며, 광선이 오브젝트에 닿지 않으면 ‘미스 셰이더’가 배경 또는 환경을 정의합니다. 셰이더 테이블이 각 사례별로 올바른 셰이더를 바인딩하여 이러한 상호 작용을 동적으로 관리합니다.

이 시스템은 복잡한 씬에는 적합하지만, 프로세싱 오버헤드를 더한다는 단점이 있으며 범용 컴퓨팅 셰이더에는 적합하지 않습니다. 간단한 효과에 동적 셰이더 테이블을 사용할 경우 불필요한 복잡성이 더해질 수 있습니다.

인라인 레이트레이싱

DXR 1.1에는 기존 레이트레이싱 파이프라인 없이 컴퓨팅 및 래스터화 셰이더 내에서 하드웨어 가속 광선 쿼리를 사용할 수 있는 인라인 레이트레이싱이 도입되었습니다.

이는 하드웨어 가속 레이트레이싱을 처리하기 위한 대안이 되며, 셰이더 및 커스텀 파이프라인 효과를 만들 새로운 가능성을 열어 줍니다.

Unity 6에서는 DXR1.1 지원 Windows 플랫폼과 콘솔을 타게팅하는 경우 이 새로운 기능이 모든 컴퓨팅 및 래스터화 셰이더 단계에서 레이트레이싱 API를 통해 완전히 지원됩니다.

셰이더에서 광선 쿼리를 사용하는 방법은 [DXR 1.1 및 RayQuery 오브젝트의 DirectX 사양](#)(영문)을 참조하세요. [인라인 레이트레이싱 지원](#)에 관한 자세한 내용은 공식 Discussions 게시물을 참조하세요.



인라인 레이트레이싱은 이와 같은 커스텀 소프트 섀도우를 만듭니다.

성능

레이트레이싱을 사용하면 성능 비용이 늘어납니다. 레이트레이싱 월드를 구축하고 업데이트할 때 GPU 사용량은 월드의 복잡도와 업데이트 주기에 따라 늘어나며, 영향을 받는 픽셀 수가 많아질수록 광선을 보내고 효과를 적용하는 작업량도 선형적으로 늘어납니다.

화질과 성능의 균형을 이루는 데 도움이 되는 몇 가지 팁은 다음과 같습니다.

- **다이나믹 해상도:** 레이트레이싱 작업량은 픽셀 수에 따라 늘어나므로 낮은 해상도로 렌더링하면 성능이 향상됩니다. 최신 업스케일러(예: NVIDIA DLSS)는 낮은 해상도로 렌더링한 다음 비주얼 충실도를 거의 손실하지 않으면서 최대 해상도까지 지능적으로 업스케일링합니다.

예를 들어, DLSS를 활성화한 다음 강제 화면 비율을 75%로 설정할 수 있습니다. 이렇게 하면 프레임 시간이 개선되며, GPU 부하를 줄이면 전체적인 성능이 향상되는 CPU 바운드 시나리오에서는 특히 더 그렇습니다.



- **효과 선별:** 모든 레이트레이싱 효과를 항상 활성화해야 하는 것은 아니며, 게임 애플리케이션의 어느 부분에 배포할지 선택할 수 있습니다.

예를 들어, 레이싱 게임에서는 플레이어의 자동차 렌더링을 개선하기 위해 레이트레이싱 반사를 추가할 수 있습니다. 오픈 월드 어드벤처 게임에서는 레이트레이싱 그림자와 전역 조명을 사용하여 환경에 한층 더 높은 몰입감을 부여할 수 있습니다.

- **필요할 때만 레이트레이싱 사용:** 레이트레이싱 반사와 전역 조명 같은 여러 효과를 조합해서 사용하려면 먼저 **Mixed 트레이싱 모드**를 사용해 보세요. Mixed 트레이싱 모드에서는 먼저 빠른 레이 마칭을 사용한 다음 필요한 때만 레이트레이싱으로 폴백합니다.

비용이 높고 오래 걸리는 광선 계산을 더 줄이려면, 레이트레이싱만큼 비용을 발생시키지 않으면서 효율적인 조명과 반사를 제공하는 대체 기법(프로브 볼륨, Sky Volume 오버라이드 또는 반사 프로브)을 사용하세요.

- **계산 최적화:**

- 일부 효과에서는 **Layer Masks**를 사용하면 레이트레이싱 효과가 불필요한 오브젝트를 계산하지 못하도록 방지할 수 있습니다.
- 품질 설정을 조정합니다. 광선 길이, 샘플 수, 해상도를 커스터마이징하여 성능과 노이즈의 균형을 이룹니다. 광선이 길수록 노이즈를 줄이기 위해 더 많은 샘플이 필요하며, 샘플 수를 줄이면 GPU 비용은 감소하지만 노이즈가 증가한다는 사실을 기억하세요.
- 반사의 경우, 오브젝트가 다른 반사 내에서 반사될 필요가 없다면 반사 횟수를 줄이세요. 필요한 경우에만 레이트레이싱을 적용할 수 있도록 평활도 파라미터를 조정하세요.
- 텍스처를 사용하는 레이트레이싱 효과의 경우, 저해상도 MIP 레벨을 가져오도록 텍스처 LOD 바이어스를 적용합니다. 이렇게 하면 GPU 메모리 사용량이 줄어들고 머티리얼 계산 중에 텍스처를 가져오는 데 드는 비용이 낮아집니다. 이로 인해 화질에 큰 영향을 주지 않으면서 전체적인 성능을 개선할 수 있습니다.

- **일반 레이트레이싱 설정:** 성능과 비주얼 품질의 균형을 맞추고, 광선을 구성하고, 가속 구조를 구축하는 방법을 설정할 수 있습니다.

성능을 분석하려면 플레이 모드에서 **렌더링 디버거(Window > Render Pipeline Debug)**를 사용합니다. 이렇게 하면 각 레이트레이싱 효과의 비용을 추적하는 일련의 마커가 제공됩니다.

CPU timings RT는 CPU가 레이트레이싱 작업을 처리하는 데 걸리는 시간(속성 바인딩과 C# 실행 포함)을 밀리초 단위로 측정합니다. **GPU timings RT**는 레이트레이싱 효과당 GPU의 실행 시간을 추적합니다.

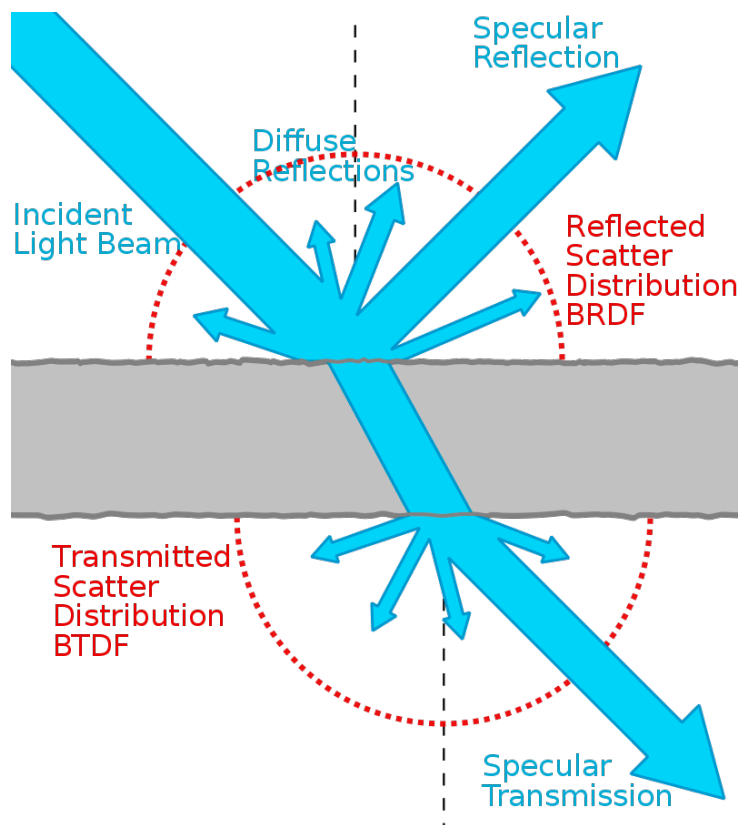
패스트레이싱

패스트레이싱은 레이트레이싱의 한 유형으로, 여러 표면에서의 반사를 포함하여 각 광선이 지나갈 수 있는 수많은 경로를 시뮬레이션합니다. 이를 통해 광원과 머티리얼 간의 더욱 복잡한 상호 작용을 캡처할 수 있습니다.

레이트레이싱과 마찬가지로 패스트레이싱도 카메라에서 썬에 광선을 보내는 것으로 시작됩니다.

하지만 광선은 처음으로 표면에 닿았을 때 멈추는 대신, 썬 내에서 계속해서 여러 번 반사되어 다양한 오브젝트와 상호 작용합니다.

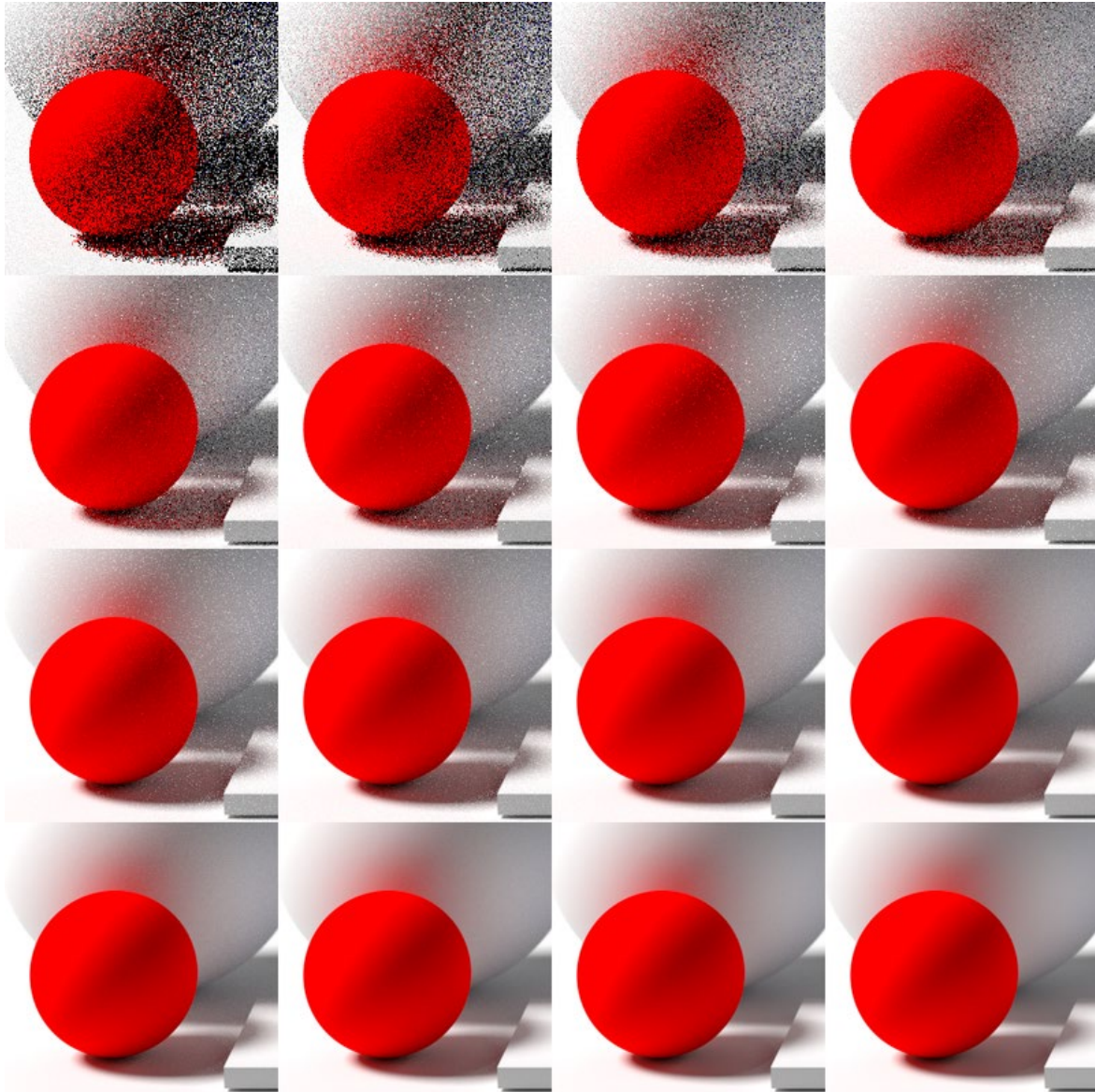
반사될 때마다 렌더러가 직접 조명과 간접 조명을 모두 반영하여 광원 정보를 수집합니다. 모든 반사 간의 색상 정보가 누적되어 픽셀의 최종 색상을 결정합니다.



패스트레이싱 산란. 출처: Wikipedia

패스트레이싱은 미묘한 난반사와 소프트 색도우를 포함한 전역 조명 효과를 더 자연스럽게 캡처합니다. 이 기술을 통해 기본 레이트레이싱에 비해 물리적으로 더 정확하며 사실적인 이미지를 얻을 수 있습니다.

그러나 패스트레이싱은 여러 번 반사되므로 레이트레이싱에 비해 계산에 많은 리소스를 소모합니다. 또한 실시간 성능을 보장하기 위해 적은 샘플을 사용할 때는 노이즈 문제가 발생할 수 있습니다.



샘플 수가 적을수록 노이즈가 발생하는 패스트레이싱. 출처: Wikipedia

이제 HDRP에는 이 문제를 완화하기 위해 다음과 같은 노이즈 제거 기술이 새로 포함됩니다.

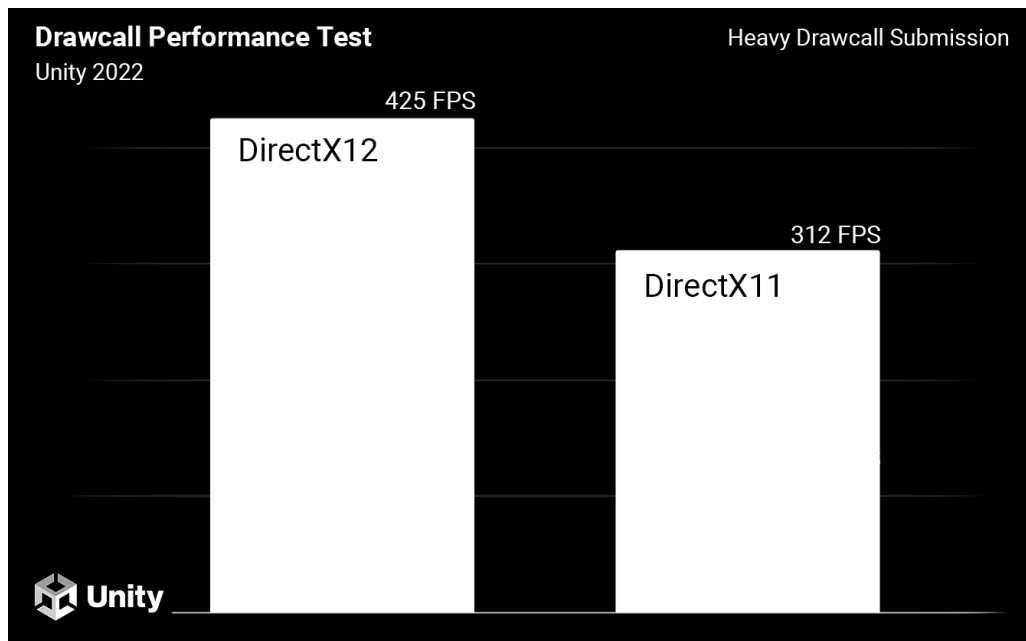
- NVIDIA Optix™ AI-Accelerated Denoiser
- Intel® Open Image Denoise(옵트인 패키지로 이용 가능)
- 또한 HDRP에서는 새로운 패스트레이싱 디노이저를 지원하기 위해 패스트레이싱을 사용하는 머티리얼 셰이더에 **Use AOVs(임의 출력 변수)** 설정이 추가됩니다. 이 설정을 활성화하면 HDRP가 알베도와 노멀 값을 AOV에 저장하여 패스트레이싱 결과를 향상합니다.

i DirectX 12

DX12(DirectX 12)는 Windows의 기본 그래픽스 API이며, 필요한 경우 DX11(DirectX 11) 폴백이 지원됩니다. Unity의 DX12(DirectX 12) 그래픽스 백엔드가 눈에 띄게 개선되었습니다. 드로우 콜이 많은 시나리오에서 DX12는 스탠드얼론 빌드에 뛰어난 CPU 성능을 보이고 있습니다.

하지만 DX12가 항상 DX11보다 우수한 결과를 보여 주는 것은 아닙니다. 예를 들어 DX11 드라이버는 DX12나 Metal, Vulkan에 비해 컴퓨트 셰이더 디스패치 호출을 더 효율적으로 재정렬할 수 있습니다. GPU 소모량이 많은 워크로드와 복잡한 컴퓨트 셰이더로 구성된 씬은 DX11에서 더 좋은 성능을 보일 수 있습니다.

에디터 성능을 개선하기 위해 DX12에는 에디터에서 [네이티브 그래픽스 잡](#)을 실행할 수 있는 옵션이 도입되었습니다. 커맨드 라인 인자 **-force-gfx-jobs native**를 사용하면 이 기능을 활성화할 수 있습니다.



Unity 2022에서 DirectX 12와 DirectX 11을 비교한 성능 테스트

- 프로젝트가 GPU 바운드인 경우 DX11을 사용하세요.
- CPU 바운드인 경우 DX12를 사용하세요.
- 에디터 성능을 최대한 활용하고 싶은 경우 에디터에서 네이티브 그래픽스 잡을 활성화하고 DX12를 사용하면 됩니다.

Unity 6의 샘플

레이트레이싱이 실제로 사용되는 사례를 보려면 Unity 6 패키지 관리자의 새로운 Transparency 샘플을 살펴보세요. 이 샘플 썬에서는 레이트레이싱 투명성, 반사 및 조명 상호 작용을 볼 수 있습니다. 레이트레이싱 효과를 활성화하여 유리, 물 및 기타 투명 머티리얼을 더 사실적으로 렌더링하세요.



Transparent 샘플에서 레이트레이싱 효과를 살펴볼 수 있습니다.

환경 조명

현실 세계에서는 빛이 여러 표면 사이에서 반사되고, 대기를 통과하면서 산란되고, 지면에 반사됩니다. 이 간접 조명은 씬의 전체적인 밝기와 색상을 정의하는 데 특정한 역할을 합니다.

HDRP에서는 환경 조명이 이 효과를 시뮬레이션하여, 직접 광원이 없는 경우에도 오브젝트가 하늘로부터 주변광을 받도록 합니다. 이는 광자가 관찰자(observer)에게 도달하기까지 대기와 땅 사이에서 무작위로 튕겨 다니면서 생기는 결과입니다.

Unity에서 하늘은 카메라가 프레임을 렌더링하기 전에 드로우하는 배경 유형입니다. 이 유형의 배경은 깊이감을 제공하고 환경이 실제보다 훨씬 크게 느껴지도록 합니다. Unity는 하늘을 사용하여 씬에서 사실적인 주변광을 생성할 수도 있습니다.

샘플 씬에서는 다른 광원이 비활성화된 경우에도 Visual Environment에 설정된 일반적인 주변광이 적용됩니다.

기본 조명 데이터 에셋

Unity 6에서는 기존의 [SkyManager](#) 시스템이 새로운 **기본 조명 데이터 에셋**으로 대체되었습니다. 이 조명 데이터 에셋은 에디터에 임베드되어 있습니다.

새 씬을 추가하면 Unity가 이 에셋을 자동으로 할당합니다. 조명 데이터 에셋은 숨겨진 [앰비언트 프로브](#)와 반사 프로브를 사용하여 기본 환경 조명을 제공하며, 프로브는 Unity 빌트인 기본 스카이박스의 환경 조명을 캡처합니다. 이 새로운 기본 에셋은 프로젝트 창에 나타나지 않습니다.



숨겨진 프로브는 업데이트되지 않으며 고정되고 미리 계산된 데이터를 포함하고 있습니다. Lighting 창에서 Skybox Material을 변경하면 이러한 프로브가 새로운 스카이박스를 캡처하지 않습니다.

기본 조명 데이터 에셋의 사용을 중단하려면 Lighting 창에서 **Generate Lighting** 또는 **Clear Baked Data**를 선택하세요. 그러면 Unity가 새 기본 조명 데이터 에셋과 새 베이킹된 반사 프로브(앰비언트 반사 프로브)를 만들고 사용합니다. 이러한 에셋은 프로젝트 창에 나타나며, Generate Lighting을 선택하면 업데이트됩니다.

예전에는 SkyManager를 통해 환경 조명이 기본적으로 씬에 영향을 미칠 수 있었으나, 이로 인해 에디터와 빌드된 플레이어 사이에 비일관성이 유발되기도 했습니다. 새로운 기본 조명 데이터 에셋은 보다 예측 가능한 워크플로를 지원합니다.



환경 조명만 사용 - 태양 역할을 하는 방향 광원을 비활성화했으나, 하늘은 여전히 주변광을 제공합니다.



주요 광원으로 해를 추가하면 씬의 일반적인 조명이 완성됩니다. 환경 조명은 그림자 영역을 채워 어두운 부분이 부자연스럽게 표현되지 않도록 합니다.



환경 조명과 결합된 태양 직사광

비주얼 환경

대부분의 경우 기본 하늘을 씬의 조명과 아트 목표에 맞는 커스텀 설정으로 대체할 것입니다.

HDRP에서 **Visual Environment** 오버라이드를 사용하면 씬의 하늘과 분위기를 정의할 수 있습니다.

Ambient Mode: Dynamic을 사용하여 하늘 조명을 Visual Environment의 **Sky > Type**에 표시된 현재 오버라이드로 설정하면 됩니다. 또는 **Ambient Mode: Static**을 사용하면 **Lighting** 창의 **Environment** 탭에 있는 하늘 설정이 기본값으로 사용됩니다.

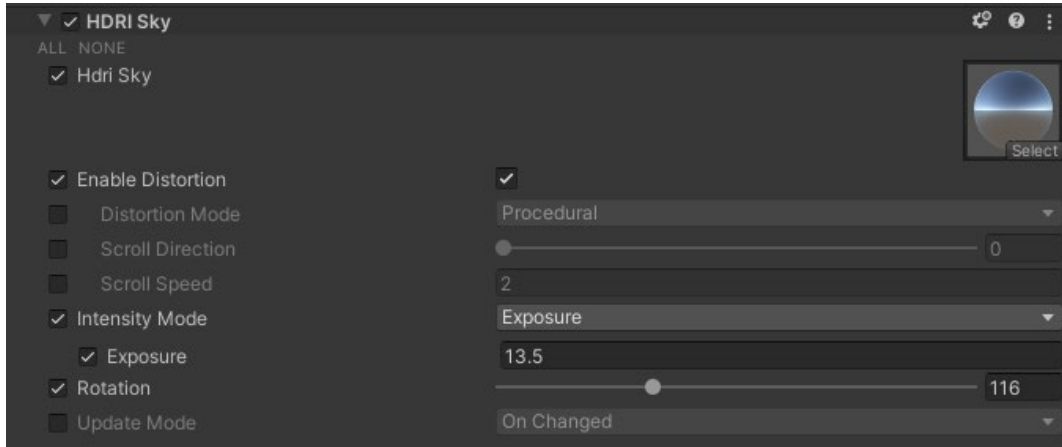
HDRP에서는 세 가지 기법으로 하늘을 만들 수 있습니다. **Type**을 **HDRI Sky**, **Gradient Sky**, **Physically Based Sky** 중 하나로 설정한 다음 Sky 메뉴에서 적절한 오버라이드를 추가하면 됩니다.

비주얼 환경 하늘을 적용하는 것은 전체 가상 월드를 거대한 조명 구체로 감싸는 것과 비슷합니다. 색상이 지정된 구체의 폴리곤은 하늘, 수평선, 땅에 일반적인 조명을 제공합니다.

HDRI Sky

HDRI Sky 오버라이드를 사용하면 **HDR 사진**으로 만든 큐브맵으로 하늘을 표현할 수 있습니다. 다양한 HDRI 소스를 인터넷에서 무료 또는 저렴한 비용으로 구할 수 있습니다. 에셋 스토어의 **Unity HDRI Pack**을 먼저 살펴보는 것도 좋습니다.

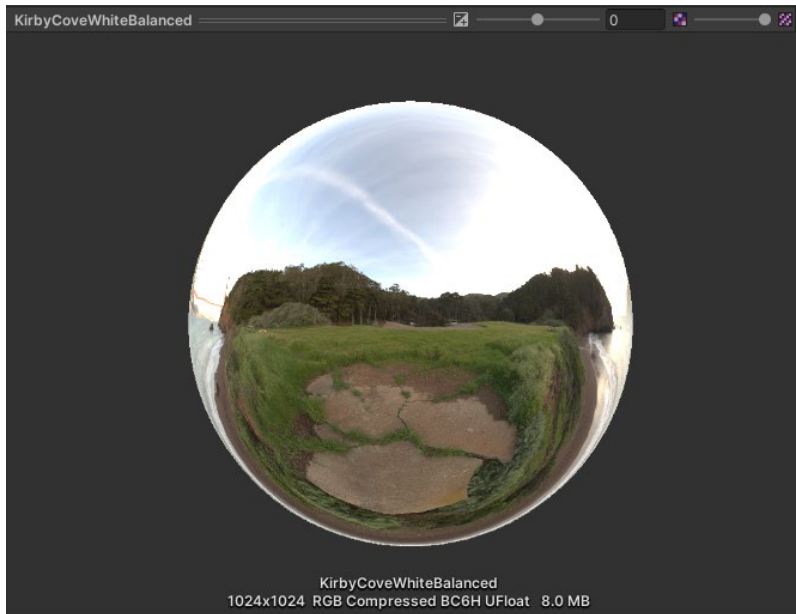
더 다양하게 실험하고 싶다면 **HDRI 촬영 가이드**를 참조하세요.



HDRI Sky 에셋

HDRI 에셋을 импорт했으면 **HDRI Sky** 오버라이드를 추가하여 **HDRI Sky** 에셋을 로드합니다. 그러면 **Distortion**, **Rotation**, **Update Mode** 등의 옵션을 조정할 수도 있습니다.

하늘이 광원이므로, **Intensity Mode**를 지정하고 해당하는 **Exposure/Multiplier/Lux** 값을 선택하여 환경 조명의 강도를 제어합니다. 강도 및 노출 값 예시는 위의 조명 및 노출 요약 자료를 참조하시기 바랍니다.



구체 내부에 큐브맵으로 적용된 HDRI Sky

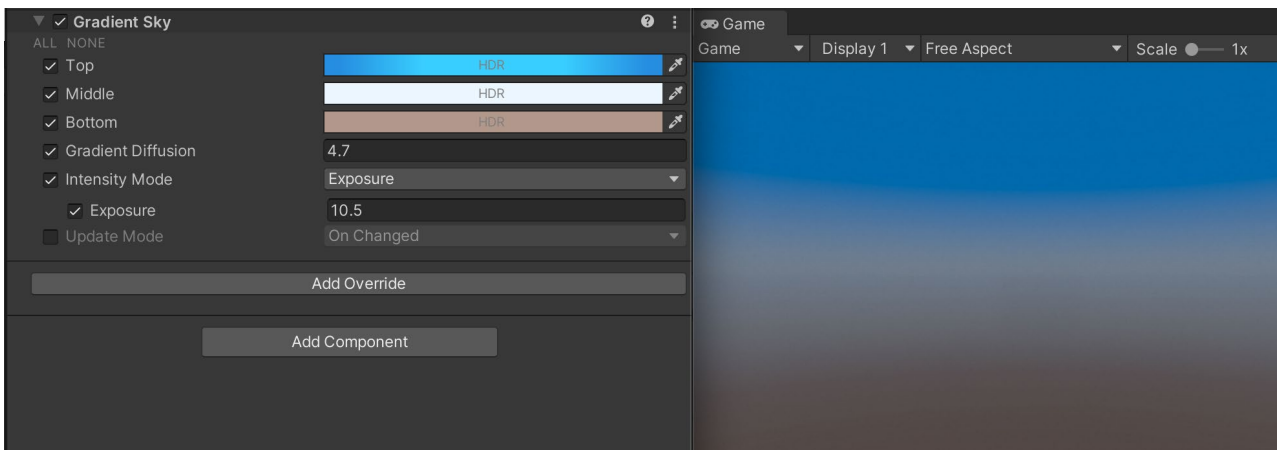
HDRI Sky 애니메이션

절차적으로 또는 플로 맵을 사용해서 HDRI 맵을 왜곡하여 **HDRI Sky**를 애니메이션화할 수 있습니다. 이를 통해 정적 HDRI에서 바람 효과를 모방하거나 더 구체적인 VFX를 만들 수 있습니다.

Gradient Sky

그라데이션을 사용해 배경 하늘을 실제와 비슷하게 만들려면 Visual Environment 오버라이드에서 **Gradient Sky**를 선택한 후, **Gradient Sky** 오버라이드를 추가합니다. **Top**, **Middle**, **Bottom** 필드를 사용해 그라데이션 색상을 정합니다.

그라데이션을 **Gradient Diffusion**으로 블렌딩하고 **Intensity**로 조명의 강도를 조절합니다.

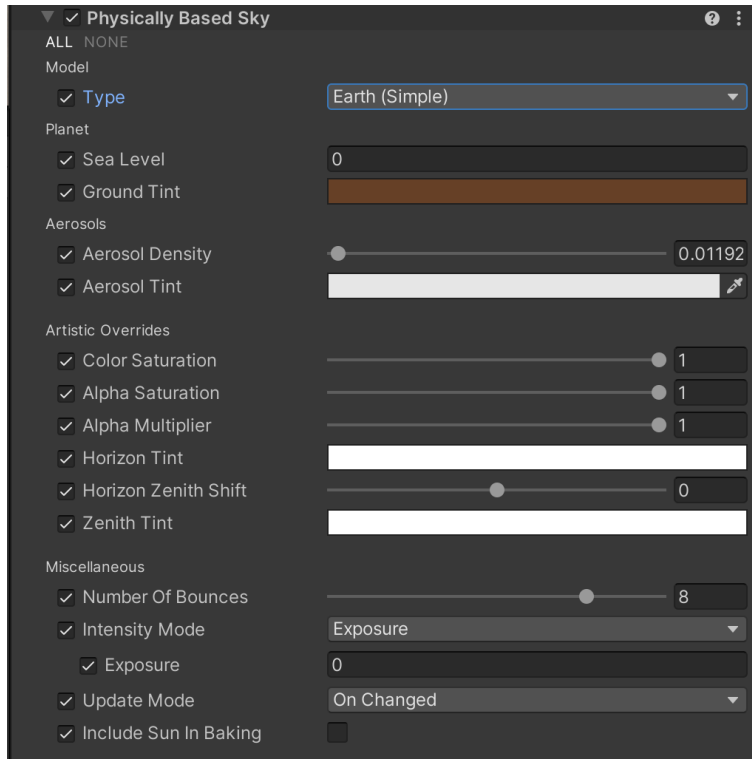


Top, Middle, Bottom 색상이 Gradient Sky에 블렌딩됩니다.

Physically Based Sky

그라데이션보다 훨씬 더 사실적인 하늘을 만들려면 **Physically Based Sky (PBR)** 오버라이드를 사용하면 됩니다.

HDRP의 Physically Based Sky 오버라이드는 공기 입자와 에어로졸 입자로 구성된 대기를 시뮬레이션합니다. 이는 **Physically Based Sky 오버라이드**에서 Volume 시스템을 사용하여 조정할 수 있습니다.



Physically Based Sky 오버라이드

이 오버라이드는 **미 산란**(Mie scattering)과 **레이일리 산란**(Rayleigh scattering) 등의 현상을 모두 포함하는 하늘을 절차적으로 생성합니다. 이렇게 하면 대기를 통해 분산되는 빛을 시뮬레이션하여 자연스러운 하늘 색상을 재현할 수 있습니다. PBR Sky로 정확하게 빛을 시뮬레이션하려면 방향 광원이 필요합니다. Physically Based Sky 오버라이드는 환경 조명 소스로도 기능할 수 있습니다.



‘시간의 유령’의 PBR 하늘



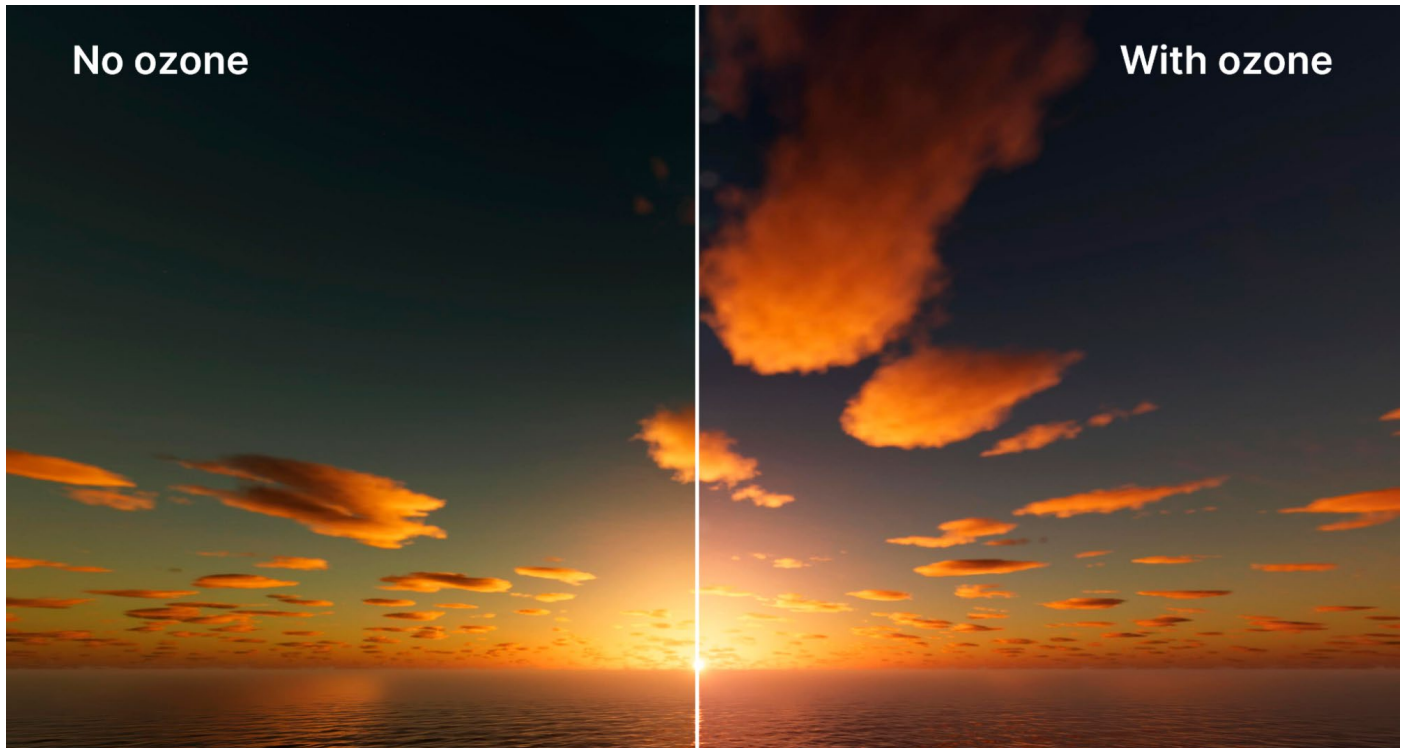
색상 팁

오브젝트가 반사 프로브의 영향을 받지 않는 실제 지면(예: 터레인)의 평균적인 색상에 따라 지면 색상을 선택하는 것이 좋습니다.

Unity 6의 PBR 하늘 업데이트

PBR 하늘의 효율이 향상되었으며, 메모리 사용량이 줄어들었고 성능이 개선되었습니다. 다중 산란에 더 이상 특정 사전 컴퓨팅 단계가 요구되지 않아 해가 수평선 아래에 있을 때 빛의 산란이 개선됩니다.

PBR 하늘의 대기 모델은 이제 오존 레이어를 포함합니다. 해가 수평선 근처에 있을 때(예: 황혼) 햇빛이 두꺼운 대기를 낮은 각도로 통과하면서 미세하게 보라색이 감도는 현상을 볼 수 있습니다.



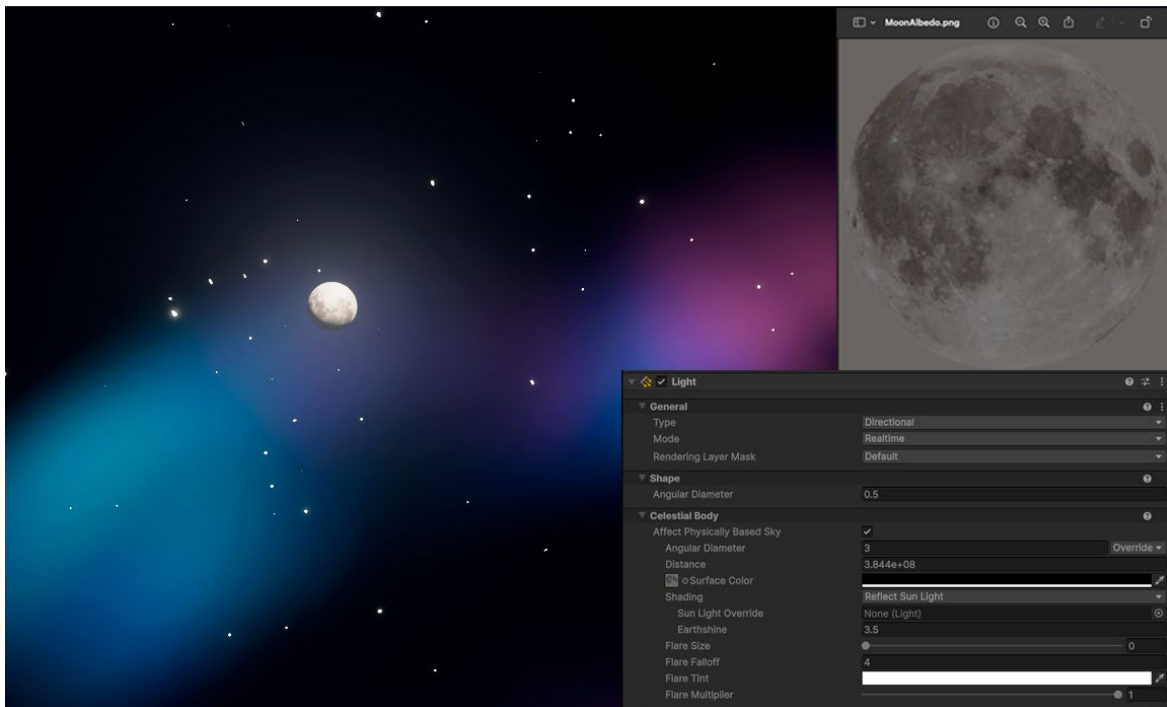
오존은 노을이 질 때 하늘에 색조를 더합니다.

이제 **항공 시점**에서 산이나 구름 같은 멀리 있는 오브젝트를 자연스럽게 페이드아웃하여 대기 중 입자에 의한 빛 흡수를 시뮬레이션합니다.

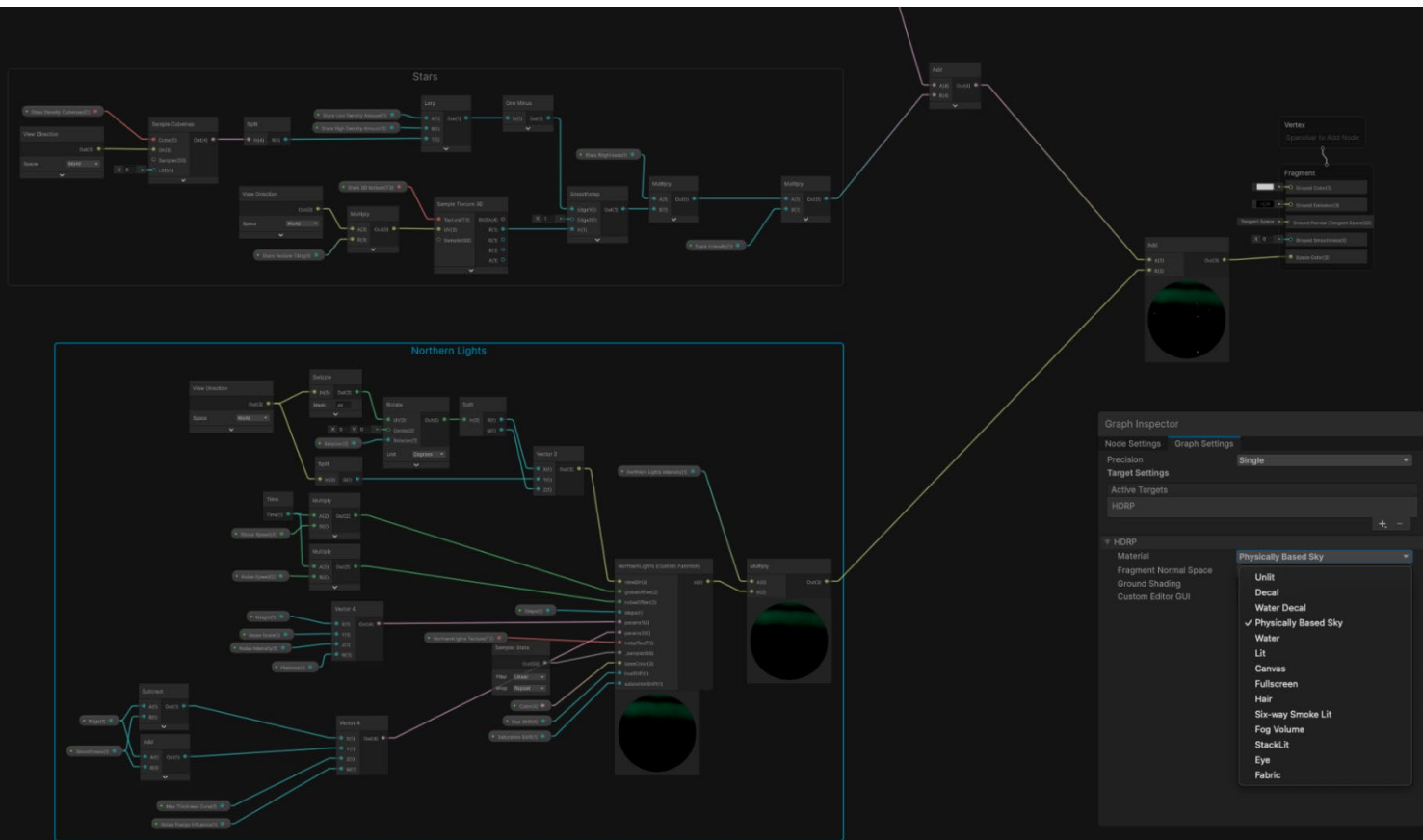


대기 산란이 구름을 더 사실적으로 렌더링합니다.

Unity 6에는 밤하늘 렌더링을 간소화하는 새로운 기능이 있습니다. Shader Graph의 **PBRSky** **머티리얼 유형**을 사용하면 절차적 별과 같은 효과를 구현할 수 있으며, 천체가 주요 방향 광원으로부터 조명을 받도록 천체를 달로 표시하는 새로운 옵션도 제공됩니다.



천체가 달을 나타낼 수 있습니다.



Shader Graph를 사용하면 절차적 별이나 오로라와 같은 효과를 구현할 수 있습니다.

대기 산란

성능 최적화를 위해 하늘에 대기 산란이 적용되어, 높은 고도에 있는 볼류메트릭 안개로 인해 발생하는 비용은 피하면서 파란색 빛이 멀리 있는 오브젝트와 클라우드에 걸쳐 산란되는 방식을 시뮬레이션합니다. 시스템은 다음과 같은 세 가지 주요 상호 작용을 모델링합니다.

- **레이리 산란**(Rayleigh scattering)은 공기 분자로 하늘의 파란색을 만듭니다.
- **미 산란**(Mie scattering)은 에어로졸로 연무, 안개, 오염과 같은 효과를 만듭니다.
- **오존 흡수**는 산란에 기여하지 않으면서 하늘의 컬러 그레디언트를 바꿉니다.

하늘 렌더링이 이제 멀리 있는 오브젝트와 구름의 대기 산란을 시뮬레이션하여 이들이 주변광과 상호 작용하는 방식을 개선합니다. 이는 하루 동안의 시간 변화를 구현하는 데 유용하며, 저녁에서 밤으로 넘어가는 황혼 무렵에 부자연스럽게 어두운 구름이 생성되는 것을 방지합니다.

그 결과 안개 거리로 큰 값을 요구하지 않으면서 조명이 더 부드럽게 전환되고 하늘이 더 자연스럽게 표현됩니다. 이에 더해 에어로졸 산란이 개선되어 더 사실적인 대기 덩스와 일관성을 지원합니다.

렌더링 공간

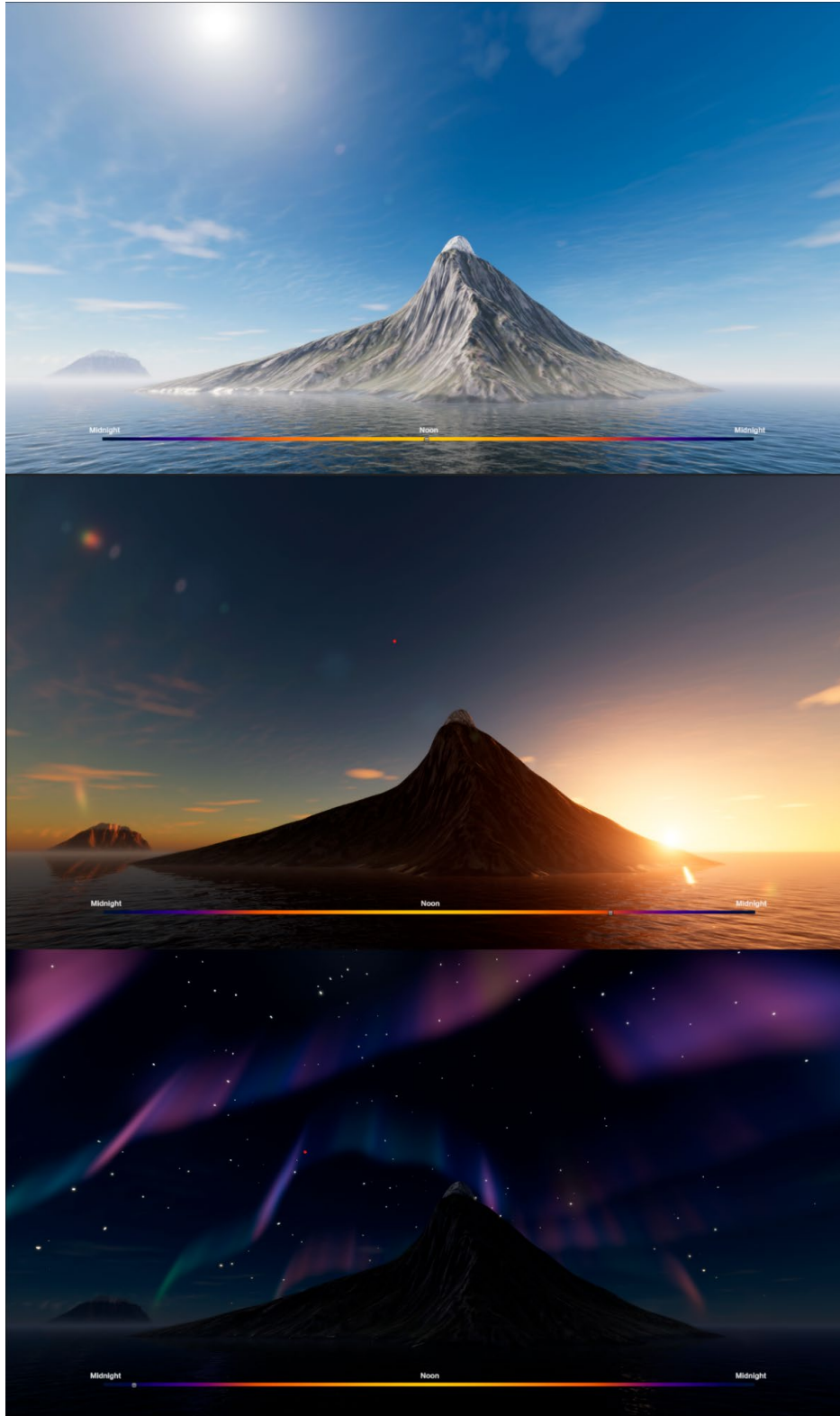
Visual Environment의 **Rendering Space** 설정은 행성과 하늘 시뮬레이션이 동작하는 방식을 결정합니다.

- **World 모드:** 행성의 표면이 월드 원점에 고정되어 있고, Radius 프로퍼티에 따라 중심이 변경됩니다. 대기 산란은 모든 가능한 광원과 관측 각도에 대해 미리 계산되어 3D 텍스처에 저장되며, Unity가 런타임에 이를 다시 샘플링합니다. 이 모드에서는 카메라가 자유롭게, 우주로까지도 이동할 수 있습니다.
- **Camera 모드:** 행성이 카메라와 함께 움직이며 표면은 항상 카메라 아래에 있습니다. 이 모드에서는 씬 전체가 아닌 특정 광원 세트에 대해 하늘 산란이 미리 계산됩니다.

행성 표면은 덤스 버퍼에서 렌더링되지 않기 때문에 렌즈 플레어를 가리지 못하고 모션 블러에서도 올바르게 동작하지 않습니다.

Environment 샘플

이러한 기능이 실제로 사용되는 예시를 보려면 **패키지 관리자**에서 **Environment 샘플**을 임포트하세요. Environment 샘플에는 자정, 정오, 황혼 사이를 오가는 시간 시스템을 사용한 하늘과 구름이 있는 섬 씬이 들어 있습니다.



Environment 샘플의 하늘 설정입니다.

Environment 샘플은 HDRP 월드 구축 톨의 터레인과 구름을 사용합니다. 이 주제에 관해서는 이어지는 챕터에서 살펴봅니다.

터레인

게임 월드는 플레이어의 발밑에서 시작됩니다. Unity의 터레인 에디터를 사용하면 규모와 관계없이 세부적이고 현실적인 랜드스케이프를 제작할 수 있습니다.

터레인 오브젝트는 평면으로 시작하지만, 지형적 특징을 갖춘 맵으로 만들 수도 있습니다. 높이 값을 부여해서 산을 만들고 계곡을 조각할 수 있으며, 타일을 상호 연결하여 가상 월드를 조립할 수 있습니다. 그다음 절차적인 방식으로, 또는 아티스트만의 감각으로 텍스처와 초목을 추가하여 랜드스케이프를 완성할 수 있습니다.

Terrain Tools 패키지를 사용하면 추가 조형 브러시와 유틸리티 모음으로 빌트인 워크플로를 한층 더 향상하여 터레인 에셋 편집 또는 설정을 자동화할 수 있습니다.



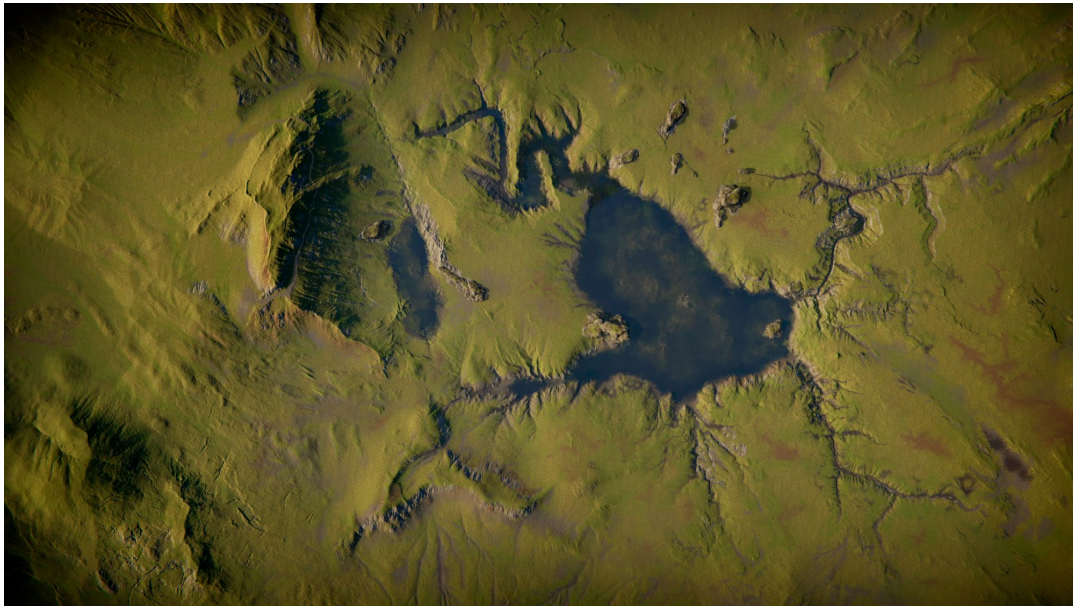
터레인 시스템은 Cloud Layer Volume 컴포넌트 오버라이드와 연동됩니다.

Unity Learn의 이 [튜토리얼 시리즈](#)를 통해 특정 톨과 기법을 사용하여 터레인을 생성하고 커스터마이징하는 방법을 알아보세요.

터레인 생성

계층 창에서 새 터레인을 생성하면 Unity가 씬에 크고 평평한 평면을 추가합니다. 여러 점으로 구성된 이 그리드는 각 버텍스별로 수정할 수 있습니다.

터레인의 모양은 그레이스케일 [하이트맵](#)에 따라 정해집니다. 이 그레이스케일 맵은 그리드의 각 버텍스를 어떻게 높이고 낮출지 결정합니다. 흰색은 가장 높은 지점을, 검은색은 가장 낮은 지점을 나타냅니다.

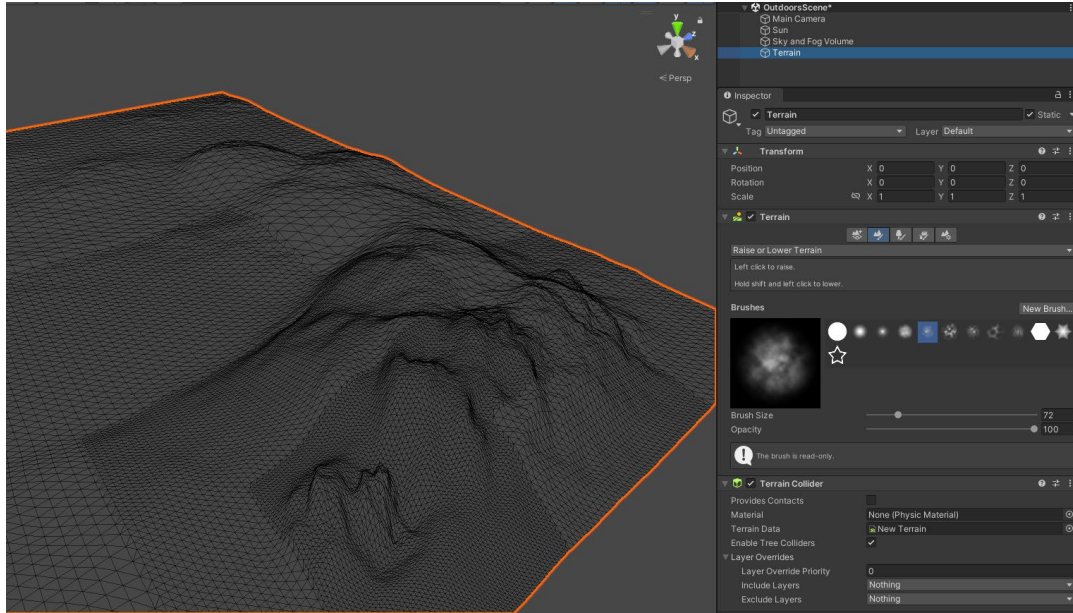


하이트맵을 칠해 터레인을 수정할 수 있습니다.

조형

Terrain 컴포넌트에는 터레인을 만들 수 있는 다양한 브러시가 포함되어 있습니다. 이러한 브러시를 사용하여 터레인의 일부를 높이거나 낮추고, 영역을 매끄럽게 만들고, 특정 셰이프나 패턴을 위한 커스텀 브러시를 만들 수도 있습니다.

Unity 6.1에서는 브러시가 실제로 사용될 때까지 전체 해상도 브러시 텍스처 생성을 지연시켜 터레인 인스펙터 성능을 개선합니다. 이로 인해 메모리 사용량이 줄어들고 터레인 인스펙터의 속도와 성능이 향상됩니다.

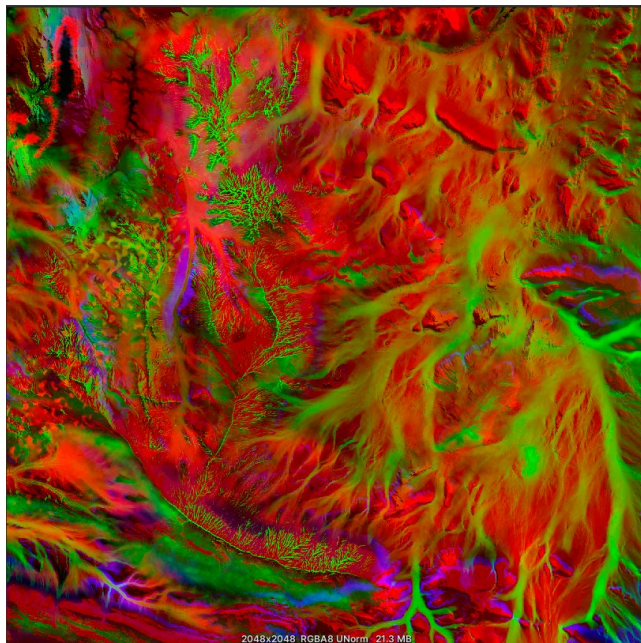


브러시를 사용하여 터레인을 만듭니다.

텍스처링 및 디테일링

[스플랫맵\(splatmap\)](#)을 사용하여 터레인에 다양한 텍스처를 칠할 수 있습니다. 스피랫맵은 이미지 편집 소프트웨어의 레이어와 같은 기능으로, 스피랫맵으로 텍스처를 정의하고 함께 블렌딩할 수 있습니다.

그다음 **디테일 맵**을 사용하여 풀, 꽃 또는 작은 돌멩이와 같은 세부적인 디테일을 더할 수 있습니다. 디테일 맵을 터레인에 직접 칠하면 됩니다.



스플랫맵을 사용하여 터레인 텍스처를 칠합니다.

나무와 초목

터레인에는 식물이 필요합니다. Unity는 Terrain 컴포넌트의 일부로 Tree Editor를 제공합니다. Tree Editor는 다양한 나무 유형과 배리에이션으로 숲과 정글을 세세하게 꾸미고 싶을 때 유용합니다.



터레인 시스템은 고급 초목 기능을 제공합니다.

Terrain 컴포넌트의 트리 팔레트를 사용하면 다양한 나이의 나무를 혼합하여 자연스러운 모습을 표현할 수 있습니다. 나무를 하나씩 배치하거나 대량 배치 툴을 사용할 수 있습니다. 그리고 나뭇잎이나 꽃 텍스처 빌보드로 디테일 레이어를 추가하면 됩니다.

나무와 초목이 바람에 흔들리도록 [Wind Zone](#) 컴포넌트를 추가하여 씬에 생동감을 더해 보세요.

SpeedTree 통합

Unity는 또한 [SpeedTree 모델](#)(.SPM 또는 .ST 파일)을 직접 임포트하여 일반적인 게임 오브젝트처럼 사용할 수 있도록 지원합니다. 터레인 에디터를 사용하여 SpeedTree 초목을 터레인에 직접 칠해 보세요. 칠한 나무는 자동으로 콜라이더를 받아 런타임 시 다른 오브젝트와 상호 작용합니다.

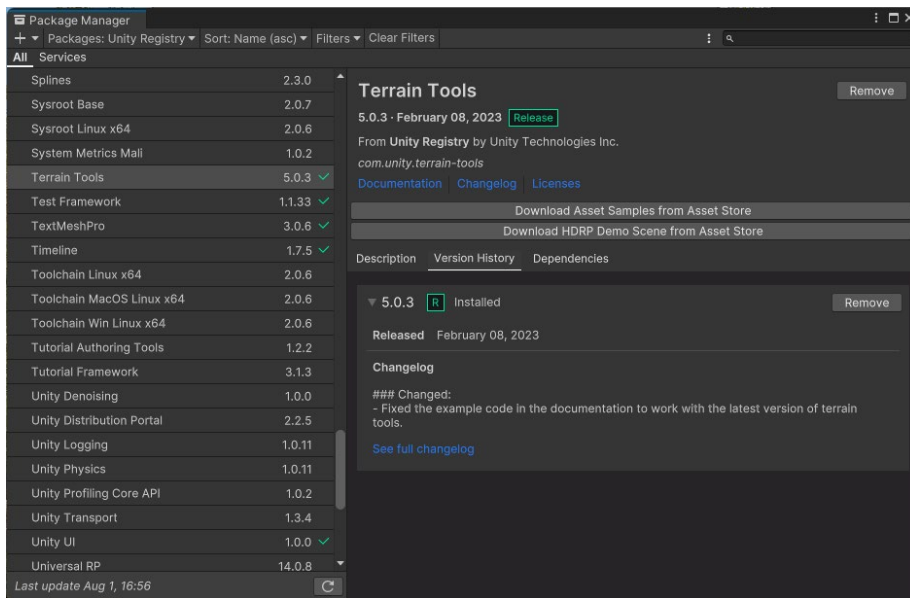
SpeedTree 모델에는 성능 최적화를 위한 빌트인 LOD가 탑재되어 있습니다. 렌더링 시스템이 효율성을 위해 모델을 배치합니다. SpeedTree 초목은 그림자를 받고 전역 조명의 영향을 받을 수 있습니다.



SpeedTree 초목

Terrain Tools 패키지

Terrain Tools 패키지는 Unity에 더 많은 터레인 조형 브러시 및 툴을 제공합니다. 애드온 툴셋을 활용하면 터레인의 표현을 더 효과적으로 제어하고 터레인 워크플로를 간소화할 수 있습니다.



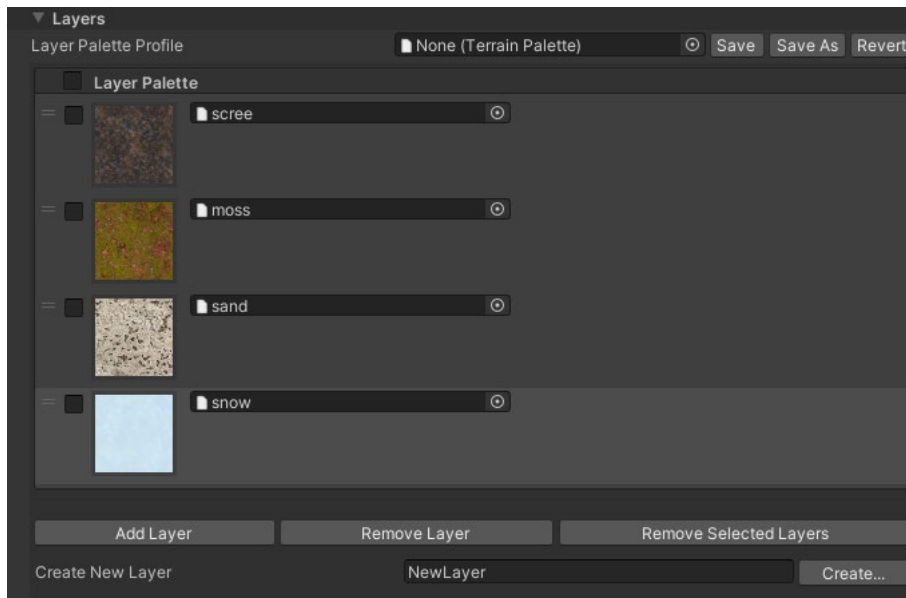
Terrain Tools 패키지를 추가하면 고급 기능을 사용할 수 있습니다.

이 패키지로 더 복잡한 형태의 터레인을 제작하거나 외부 디지털 콘텐츠 제작 툴에서 터레인 텍스처 데이터를 저작(author)할 수 있습니다.

자세히 알아보려면 [Terrain Tools 시작하기](#)를 참고하세요.

터레인 페인팅

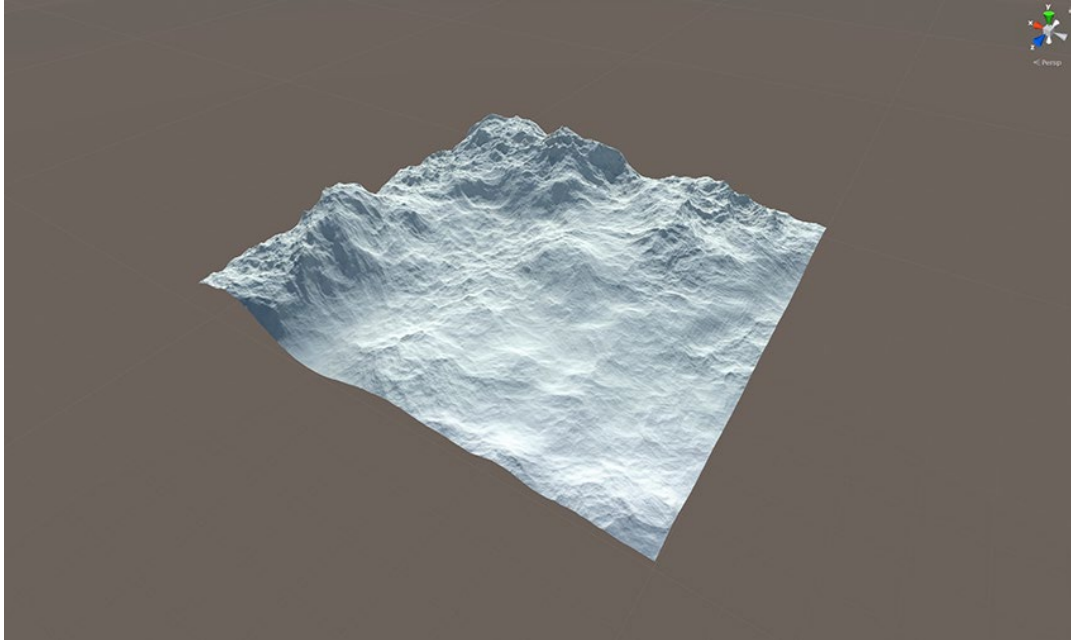
Terrain Tools 패키지는 기존 빌트인 툴에 새로운 툴을 추가하고 빌트인 [Paint Texture](#), [Smooth Height](#), [Stamp Terrain](#) 툴의 기능을 개선합니다. 터레인 페인팅 툴의 전체 목록을 살펴보면 이 [기술 자료 페이지](#)를 참고하세요.



Terrain Tools를 사용하여 레이어 단위로 칠할 수 있습니다.

Sculpt 툴로 터레인의 모양을 변경할 수 있으며, 지형을 더하거나 빼는 다음과 같은 기능이 제공됩니다.

- [Bridge](#)는 선택한 두 점 사이에 브러시 스트로크를 만들어 지형을 연결합니다.
- [Clone](#)은 한 영역의 터레인을 다른 곳으로 복제합니다.
- [Noise](#)는 다양한 유형의 노이즈와 프랙탈(fractal)을 사용하여 터레인 높이를 변경합니다.
- [Terrace](#)는 터레인을 계단식 평면으로 변환합니다.



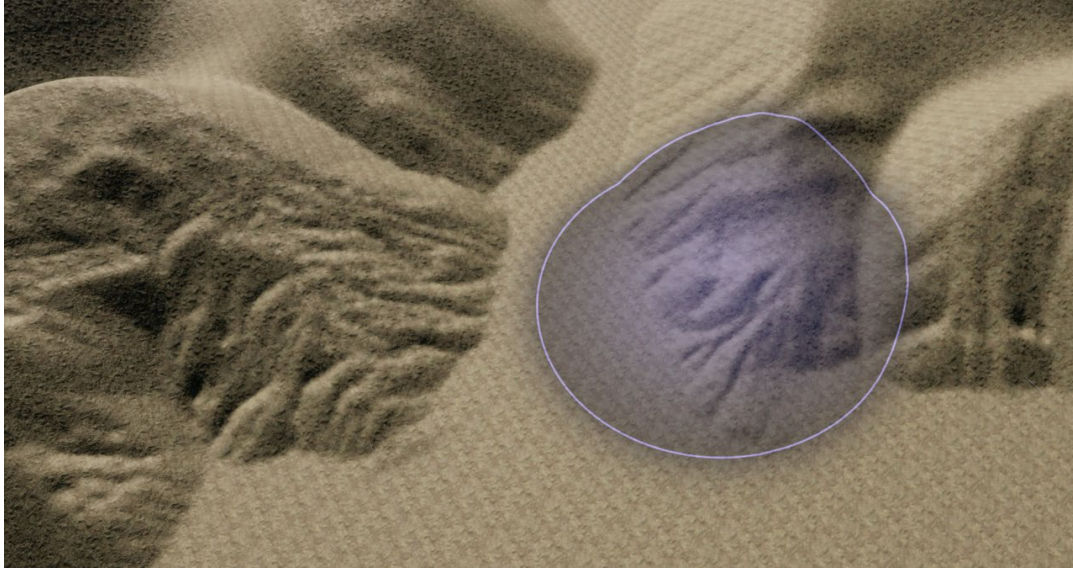
노이즈는 터레인 높이를 수정합니다.

Effects 툴은 기존 높이를 기반으로 터레인을 수정합니다.

- **Contrast**는 터레인 높이의 전반적인 범위를 늘리거나 줄입니다.
- **Sharpen Peaks**는 봉우리를 뾰족하게 만들거나 터레인의 평평한 부분을 더 평평하게 만듭니다.
- **Slope Flatten**은 평균 경사도를 유지하며 터레인을 평평하게 만듭니다.

Erosion 툴은 흐르는 물이나 바람의 영향과 침전물의 이동을 시뮬레이션합니다.

- **Hydraulic**은 물에 의한 침식으로 흐름에 따라 침전물이 이동하는 것을 시뮬레이션합니다. 이 툴을 사용하여 골짜기나 하천을 구현할 수 있습니다.
- **Thermal**은 자연스러운 경사를 유지하면서 터레인에 쌓이는 침전물의 영향을 시뮬레이션합니다.
- **Wind**는 바람으로 인한 침식의 영향과 침전물의 재분포를 시뮬레이션합니다.



Erosion 툴은 흐르는 바람이나 물의 영향과 침전물의 이동을 시뮬레이션합니다.

Transform 툴은 터레인을 밀고, 당기고, 회전시킵니다.

- **Pinch**는 브러시 중앙으로 높이를 당기거나 바깥으로 밀어냅니다.
- **Smudge**는 브러시 스트로크의 경로를 따라 터레인 지형을 이동시킵니다.
- **Twist**는 브러시 중앙을 중심으로 브러시 스트로크의 경로를 따라 터레인 지형을 회전시킵니다.



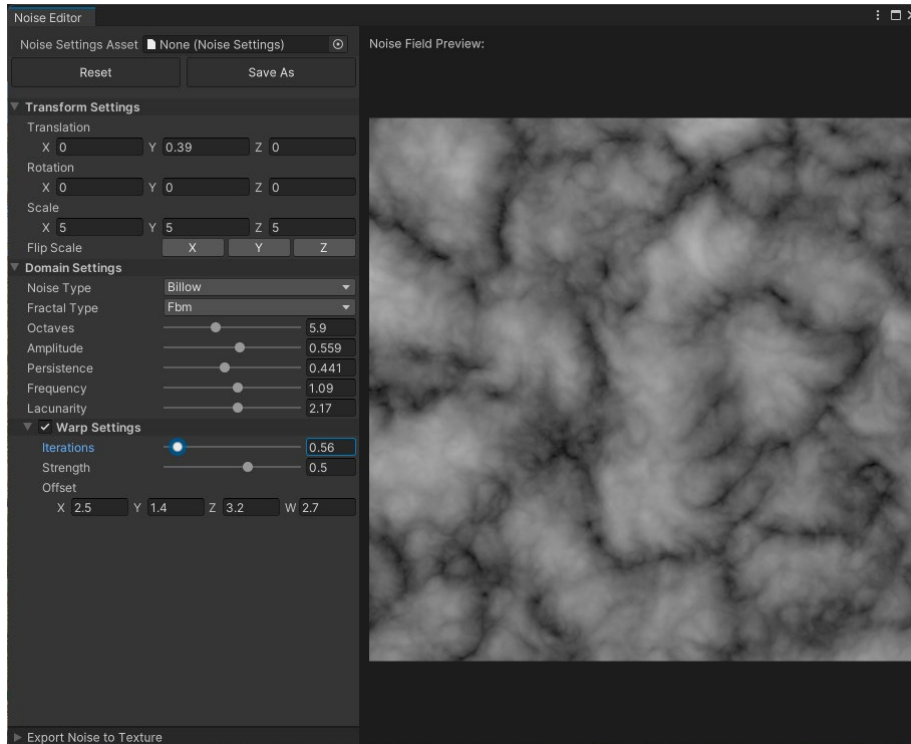
Twist는 브러시 중앙을 중심으로 터레인 지형을 회전시킵니다.

커스텀 터레인 페인팅 툴을 제작할 수도 있습니다. 자세히 알아보려면 [TerrainAPI.TerrainPaintTool](#)과 [커스텀 터레인 툴 제작](#) 페이지를 참고하세요.

Noise Editor

Noise Editor를 통해 [Noise Height Tool](#)과 [Noise Brush Mask Filters](#)에서 사용할 수 있는 노이즈 설정 에셋을 저작하고 관리할 수 있습니다. 또한 Noise Editor를 사용하여 외부에서 사용할 절차적 텍스처를 생성할 수도 있습니다.

Noise Editor를 열려면 메뉴에서 **Window > Terrain > Edit Noise**를 선택합니다.

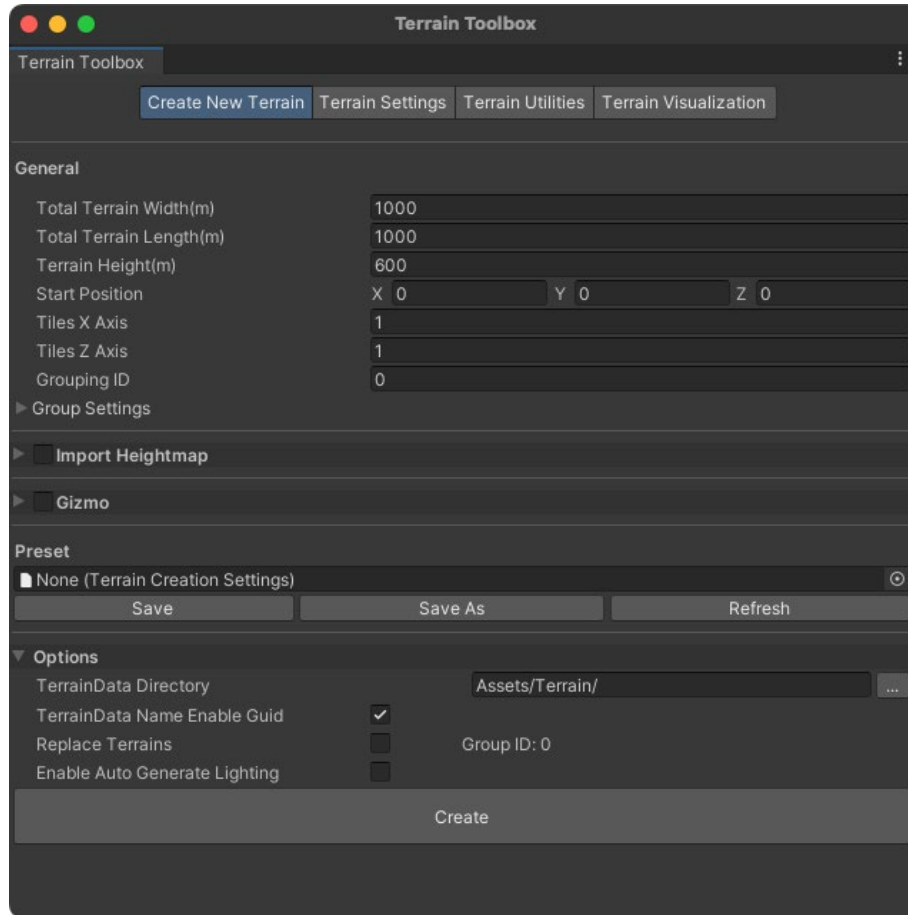


Noise Editor

Terrain Toolbox

Terrain Toolbox는 터레인 워크플로를 간소화할 수 있는 유용한 툴이 담긴 에디터 창입니다. 프리셋 설정이나 임포트된 맵에서 새 터레인을 생성하거나, 터레인 설정을 일괄 변경하거나, 스피랫맵과 하이트맵을 임포트/익스포트할 수 있습니다.

Terrain Toolbox를 열려면 **Window > Terrain > Terrain Toolbox**를 선택합니다.



Terrain Toolbox는 워크플로를 개선합니다.

터레인용 레이트레이싱 지원

Unity는 일부 제한 사항과 함께 터레인 레이트레이싱을 지원합니다. 터레인은 레이트레이싱 효과의 영향을 받을 수는 있지만, 반대로 영향을 주지는 않습니다.

터레인은 레이트레이싱 반사에서 제외됩니다. 호수 표면과 같이 필요한 경우에는 대신 평면 반사를 사용해야 합니다. 또한 레이트레이싱 앰비언트 오클루전과 전역 조명은 터레인에 영향을 주지 않습니다.

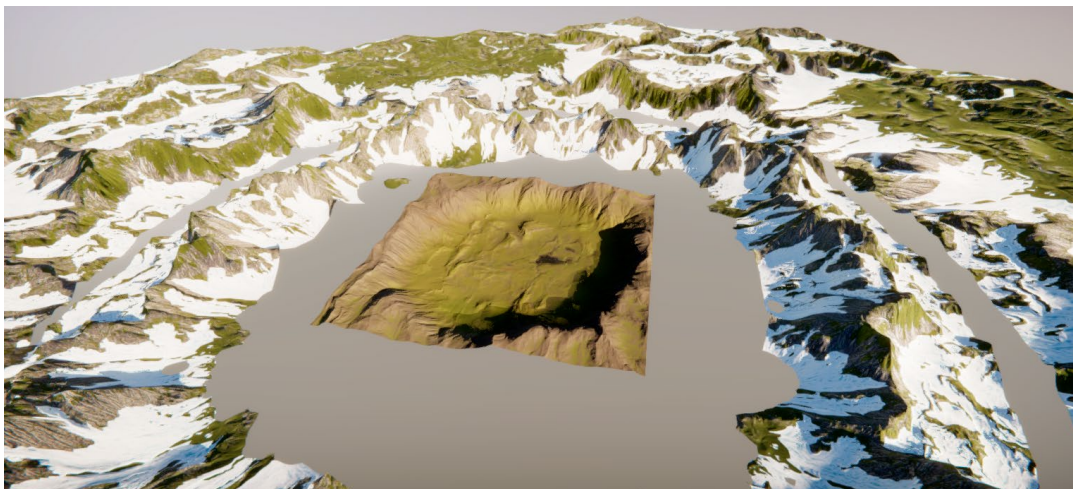
HDRP Terrain 데모

이 패키지는 [Unity 에셋 스토어](#)에서 받을 수 있으며 Unity 터레인 시스템을 최대한 활용할 수 있도록 돕는 다양한 예시 콘텐츠가 들어 있습니다. 이 샘플 팩의 텍스처, 브러시, 모델은 다른 프로젝트에도 사용할 수 있습니다.



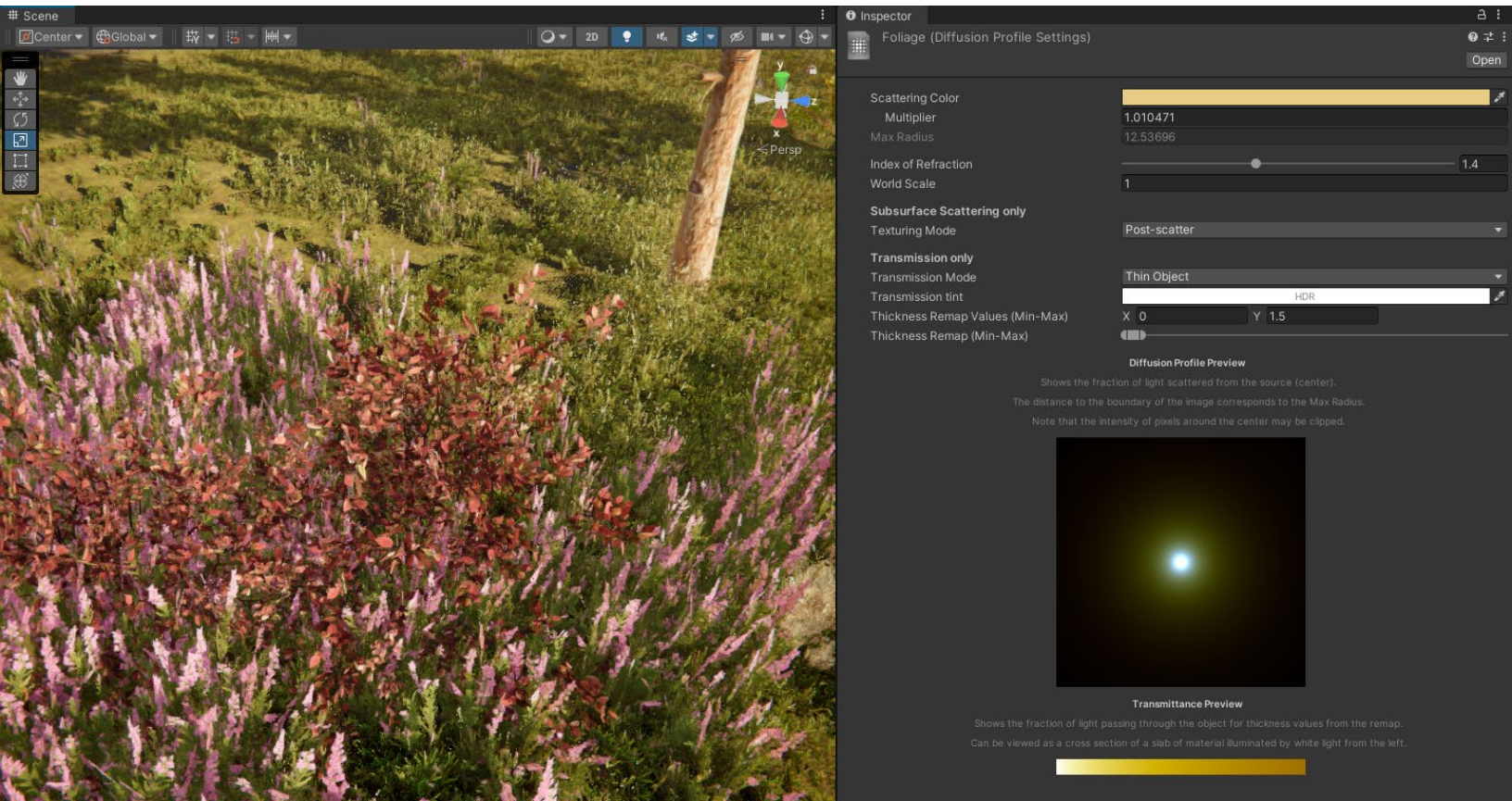
HDRP Terrain 데모는 패키지 관리자에서 다운로드할 수 있습니다.

이 씬에는 메모, 팁과 함께 여러 위치가 즐겨찾기로 추가되어 있습니다. 인스펙터의 버튼을 사용하여 원하는 지점에 카메라의 포커스를 맞추고 즐겨찾기를 바꿔 가며 확인해 보세요.



HDRP Terrain 데모의 터레인 타일

이 씬에 사용된 나뭇잎 셰이더는 피하 산란에 [HDRP Diffusion Profile](#) 에셋을 사용합니다. 그 결과 반투명한 유기적 머티리얼이 거친 플라스틱과 같은 질감 대신 부드럽고 자연스러운 느낌을 냅니다.



확산 프로파일에는 피하 산란 설정이 저장됩니다.

구름

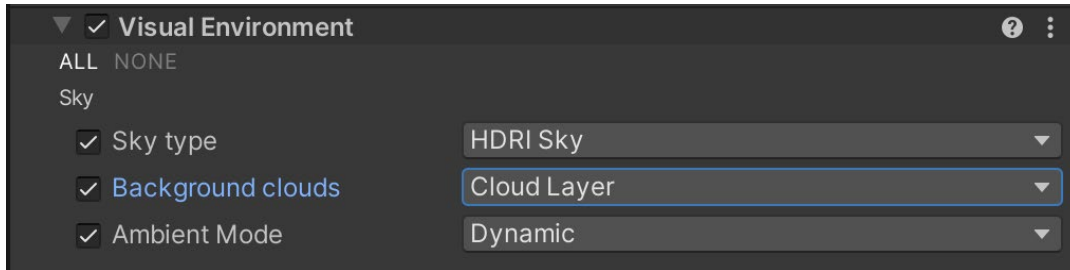
구름이 없는 하늘은 무언가 부족해 보입니다. **Cloud Layer** Volume 컴포넌트 오버라이드로 자연스러워 보이는 구름을 만들어 **Sky** 오버라이드와 **Visual Environment** 오버라이드를 보완할 수 있습니다. **볼류메트릭 구름**은 조명과 바람에 반응하며 실제 두께를 지닌 사실적인 구름을 생성합니다.



HDRI Sky 앞에 구름 레이어가 나타납니다.

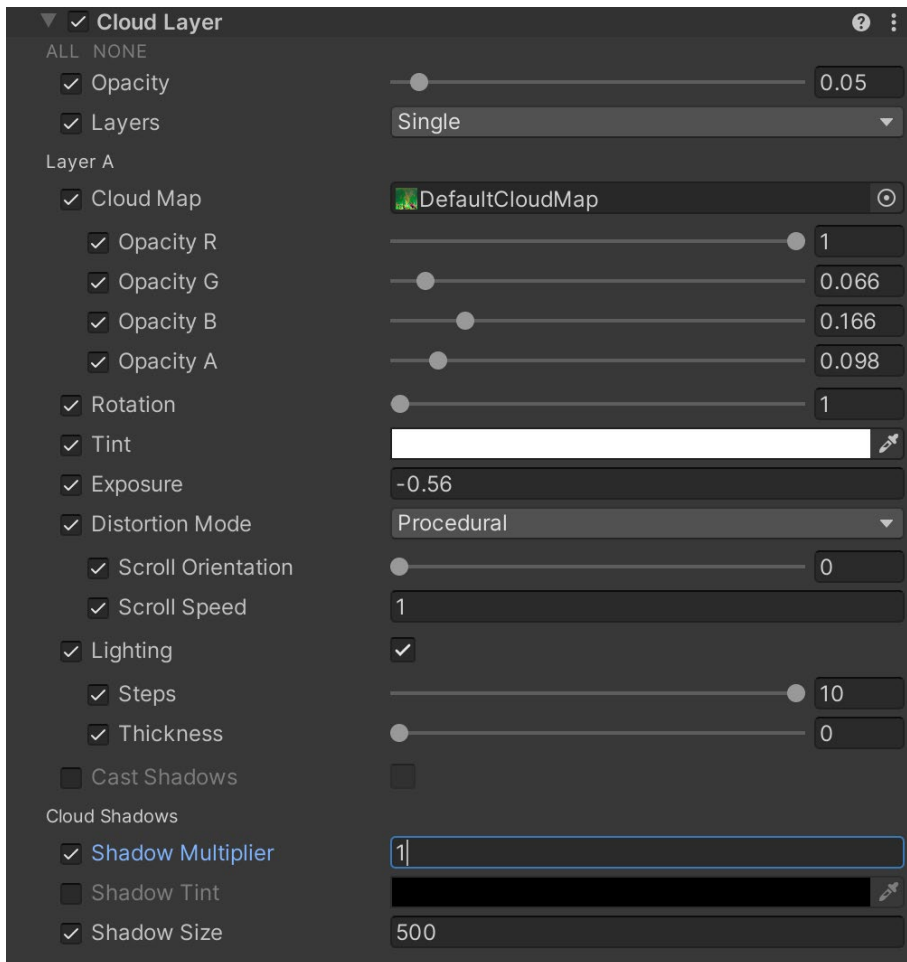
구름 레이어

Cloud Layer 오버라이드는 **플로맵**으로 애니메이션화가 가능한 2D 텍스처로, 빨간색 채널과 초록색 채널을 사용하여 벡터 변위를 제어합니다. 플레이 모드에서 구름은 하늘에 약간의 모션을 추가하여 배경을 더 역동적으로 만듭니다. 구름 레이어는 하늘 앞에 위치하며, 지면에 그림자를 드리우는 옵션을 사용할 수 있습니다.



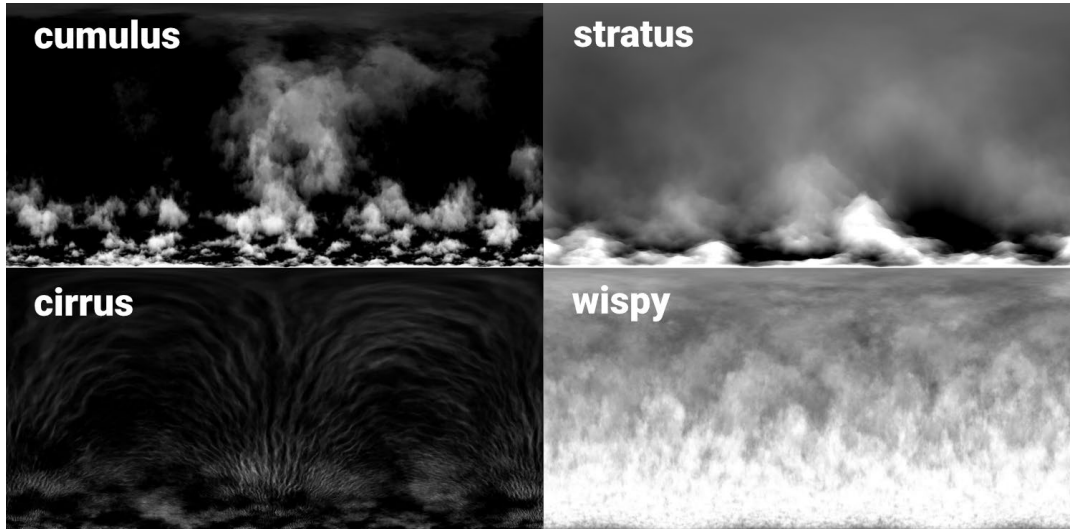
Visual Environment 오버라이드에서 Cloud Layer 오버라이드를 활성화합니다.

로컬 볼륨이나 글로벌 볼륨에서 Visual Environment 오버라이드의 **Background clouds**를 활성화한 다음 **Cloud Layer** 오버라이드를 추가합니다.



Cloud Layer 오버라이드

구름 맵 자체는 **원통 투영**을 사용하는 텍스처로, RGBA 채널 모두에 다른 구름 텍스처(각각 적운, 층운, 권운, 성긴 구름)가 있습니다. 구름 레이어를 제어하여 각 채널을 블렌딩하고 원하는 구름 모양을 만들 수 있습니다. 네 개의 채널이 있는 두 개의 레이어를 사용하면 최대 8개의 구름 풍경을 시뮬레이션하고 블렌딩할 수 있습니다.



DefaultCloudMap의 네 가지 채널

구름의 애니메이션, 조명, 색상, 그림자를 조정하여 원하는 하늘을 만들어 보세요.

i 대기 및 태양 기반 조명

Unity 6에서는 구름 레이어의 물리 기반 렌더링이 개선되었습니다.

이제 구름 레이어를 Physically Based Sky 오버라이드와 함께 사용할 때 햇빛의 색상에 대기 감쇠가 올바르게 반영됩니다.

또한 이제 레이 마칭이 비활성화되더라도 햇빛의 색상이 항상 구름의 색상에 영향을 미칩니다. 그리고 레이 마칭 알고리즘이 개선되어 산란 효과가 향상되었으며, 이제 단계 수를 변경하더라도 HDRP가 더 일관된 결과를 보여 줍니다.



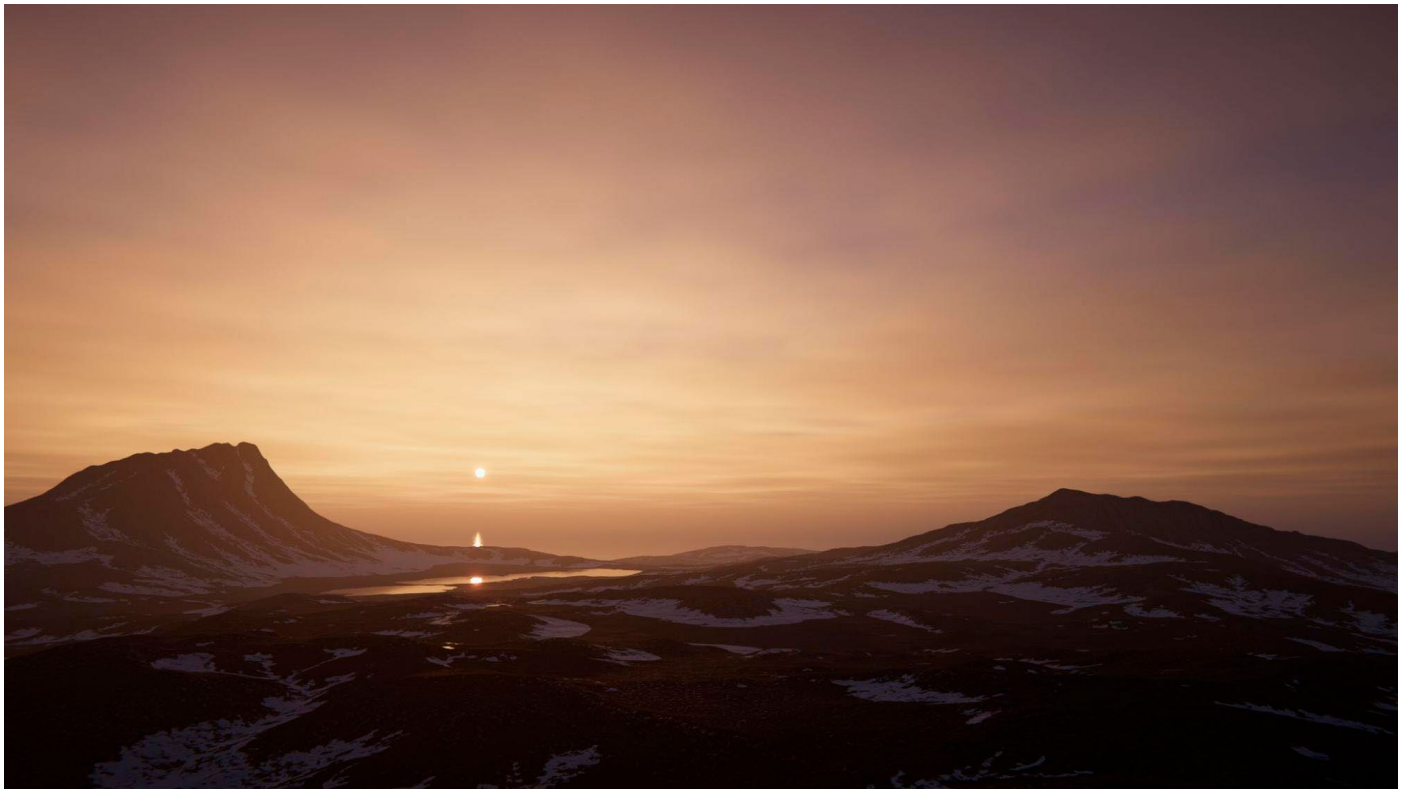
Physically Based Sky 오버라이드가 적용된 구름 레이어

볼류메트릭 구름

구름이 광원과 상호 작용해야 하는 경우, **Volumetric Clouds** 오버라이드를 사용하면 됩니다. 볼류메트릭 구름은 그림자를 렌더링하거나 안개와 섞이고, 입체적인 빛줄기를 만들 수 있습니다. 볼류메트릭 구름을 구름 레이어와 함께 사용하거나 따로 추가하세요.

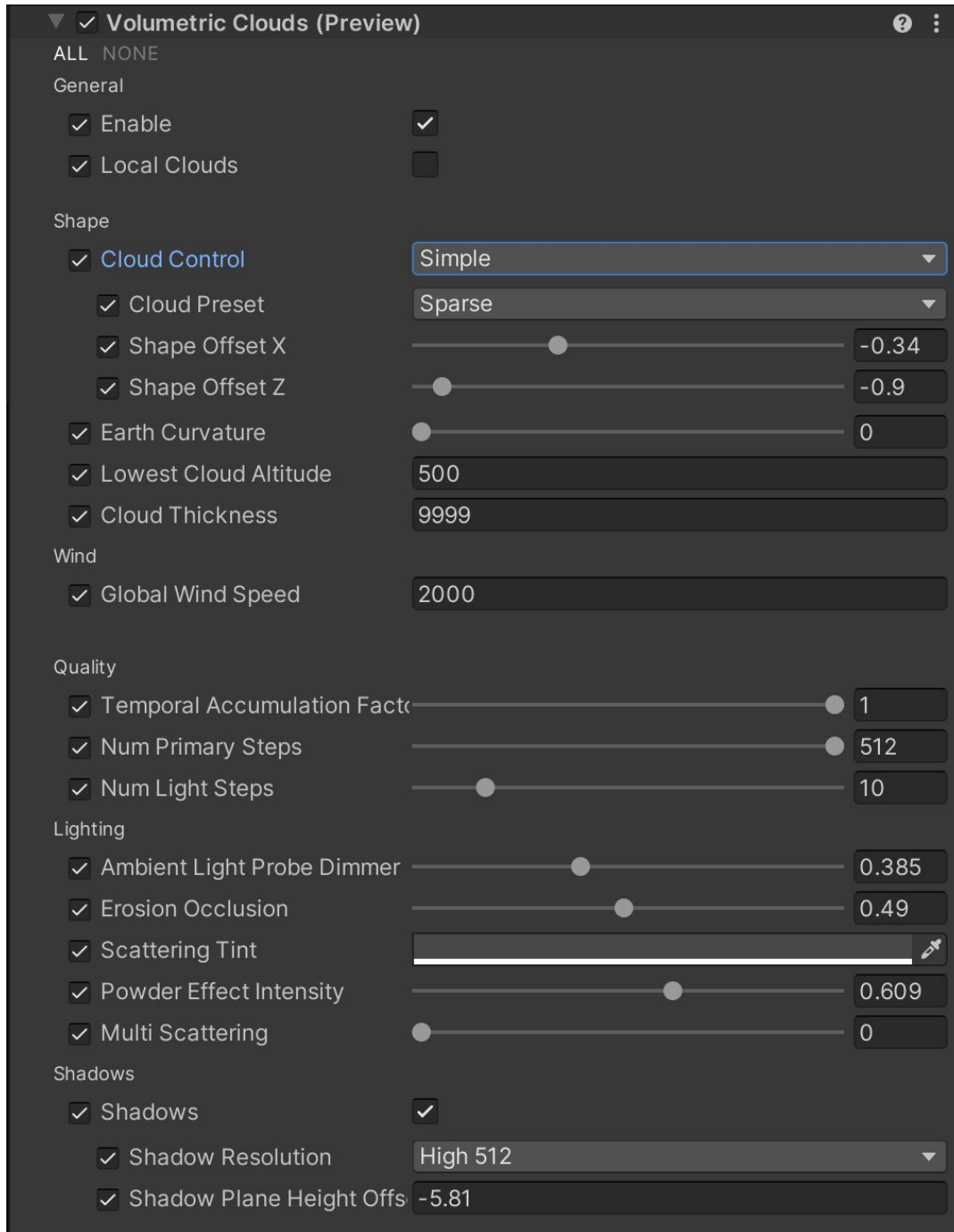
다음과 같이 Volumetric Clouds 오버라이드를 활성화할 수 있습니다.

- HDRP 에셋에서 **Lighting > Volumetric Clouds > Volumetric Clouds**를 활성화합니다.
- 로컬 또는 글로벌 볼륨에서 **Volumetric Clouds** 오버라이드를 추가합니다.



볼류메트릭 구름은 메인 방향 광원에 반응합니다.

Cloud Control의 Advanced 옵션과 Manual 옵션을 사용하면 각 구름 유형에 대한 맵을 정의할 수 있습니다.



Volumetric Clouds 오버라이드

Unity 6의 업데이트

Unity 6에서는 볼류메트릭 구름에 다음과 같은 업데이트가 적용되었습니다.

- 볼류메트릭 구름이 더 이상 원거리 평면에 의해 잘리지 않으며 카메라 거리와 관계없이 완전히 렌더링됩니다.
- 행성 설정이 **Visual Environment** 오버라이드의 일환으로 관리되어 별도의 지구 컨트롤이 필요하지 않습니다. 이는 기존 Local Clouds 옵션을 대체합니다.
- 향상된 대기 산란이 원거리 구름의 조명을 개선합니다. 이로 인해 지금까지 성능에 영향을 미쳤던 먼 안개 거리의 필요성이 줄어들었습니다.
- 볼류메트릭 구름은 향상된 새도우 맵 통합을 통해 개선된 셀프 세도잉으로 렌더링됩니다. 따라서 더 상세하고 사실적인 구름이 형성됩니다.
- 업데이트된 구름 레이어 시스템은 서로 다른 두 구름 설정 간의 블렌딩을 지원하여 부드러운 전환을 가능하게 합니다(예: 볼륨 블렌딩을 통해 구름이 드문 하늘에서 흐린 하늘로 전환). 볼륨 프레임워크의 커브 블렌딩을 사용하여 볼류메트릭 구름 간에 매끄럽게 전환할 수 있습니다.
- 이 업데이트를 통해 록업 테이블을 사용하는 이전 프리셋에서 볼 수 있었던 깜빡임이나 쉬머링(shimmering) 같은 시각적 결함이 줄어들었습니다.
- 볼류메트릭 구름은 **Visual Environment**의 **Rendering Space** 설정을 따릅니다. 카메라가 구름에 들어가지 않도록 하려면 **Camera Space**를 사용하세요.

[Cloud Layer](#) 오버라이드와 [Volumetric Clouds](#) 오버라이드에 대한 내용은 [HDRP](#) 기술 자료를 참고하세요.

Environment 샘플

패키지 관리자 Samples 탭의 [Environment](#) 샘플에는 여러 샘플 구름이 있습니다. 이러한 샘플 구름은 다양한 구름을 만들기 위한 시작점으로 활용할 수 있습니다.

이 샘플에는 다양한 **Volumetric Cloud** 설정이 있습니다. 프리팹별로 서로 다른 단계를 통해 구름 구조를 정의하는 모습을 확인할 수 있습니다.

Simple 모드: 미리 정의된 밀도 및 침식 설정을 사용하여 일반적인 구름 유형을 생성합니다. 파라미터를 조정하면 커스텀 텍스처를 사용하지 않고도 구름 범위와 모양을 수정할 수 있습니다.

Advanced 모드: 특정 월드 공간 지점에 미리 정의된 구름 유형(Cumulus, Altostratus, Cumulonimbus)을 배치합니다. 절차적으로 생성된 **Cloud LUT**가 구름의 모양, 고도, 밀도를 정의합니다.

Manual 모드: 특정 구름 유형으로 **Cloud LUT**를 정의하여 완전히 커스터마이징할 수 있습니다. **Cloud Map**은 블루 채널에서 여러 색상을 샘플링하여 구름을 배치할 지점을 정합니다. 인스펙터 UI에서 사용 가능한 프리팹을 살펴보세요.

Simple - Clear Skies: 밀도가 낮은 구름 프리셋에 열은 층운이 있습니다.



Simple Clear Skies

Simple - Cloudy: 구름의 밀도를 높이기 위해 구름 층을 추가하는 프리셋입니다.



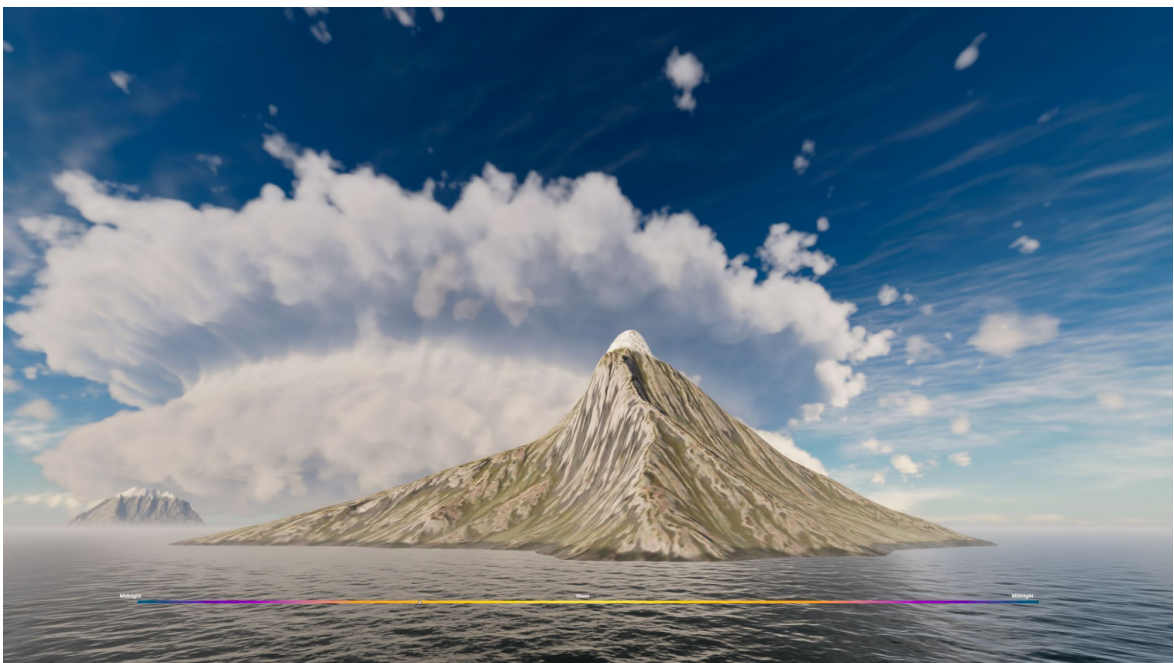
Simple Cloudy

Advanced Clouds: 섬 위에 적운을 배치하고 서로 다른 고도에 층운과 적란운을 층층이 배치합니다.



Advanced Clouds

Manual - Cumulonimbus: 이 프리팹은 커스텀 **Cloud LUT**로 적란운 하나를 배치하여 극적인 석양 효과를 연출합니다.



Manual Cumulonimbus

Manual - Cirro Stratus: 변형된 그레디언트와 노이즈 텍스처를 갖는 권층운을 배치하여 자연적인 배리어이션 효과를 연출합니다.



Manual - Cirro Stratus

Manual - Complete LUT: Cloud Map을 통해 여러 구름 유형을 샘플링한 전체 **Cloud LUT**로, 그레디언트와 노이즈 텍스처를 사용해 다양성을 부여합니다.



Manual - Complete LUT

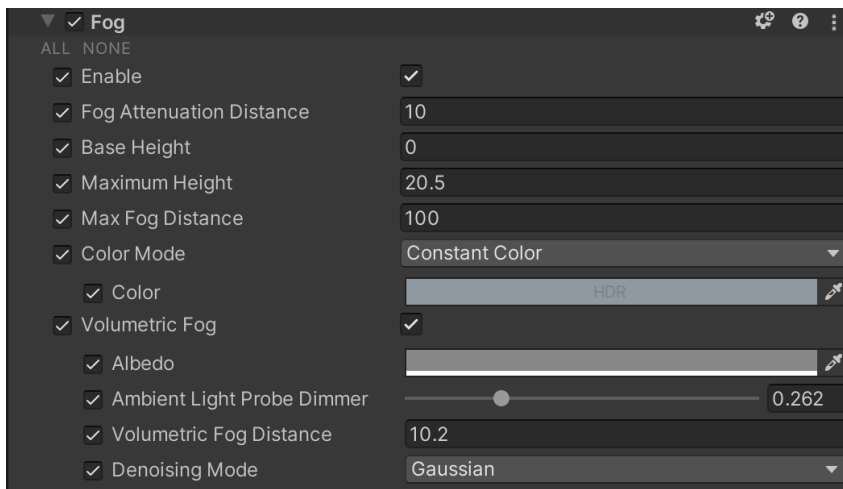
안개 및 대기 산란

연기, 안개, 아지랑이 효과는 예전부터 촬영에 많이 사용되었으며, 스테이지 조명에 깊이와 입체감을 더하거나 분위기를 조성하는 데 도움이 됩니다. HDRP에서는 안개를 사용해 비슷한 효과를 낼 수 있습니다.

안개의 불투명도는 카메라에서 오브젝트까지의 거리에 따라 달라집니다. 안개는 카메라의 원거리 클리핑 평면을 숨길 수 있어서 원거리 지오메트리를 씬에 다시 블렌딩할 수 있습니다.

글로벌 안개

HDRP는 글로벌 안개를 **Fog** 오버라이드로 구현합니다. 여기서 안개는 카메라로부터의 거리와 월드 공간 높이에 따라 기하급수적으로 열어집니다.



Fog 오버라이드



씬의 볼륨에 Fog 오버라이드를 설정합니다. **Base Height**는 일정하고 두꺼운 안개가 위쪽으로 갈수록 없어지기 시작하는 경계를 정합니다. 여기서 정한 값을 넘어가면 **Maximum Height**에 도달할 때까지 안개 밀도가 기하급수적으로 없어집니다.



Base Height와 Maximum Height 설정을 사용해 낮게 깔린 안개를 만듭니다.

마찬가지로 **Fog Attenuation Distance**와 **Max Fog Distance**를 사용하면 카메라에서 멀어질수록 안개가 어떻게 없어질지 제어할 수 있습니다. **Color Mode**를 **Constant Color**나 기존 **Sky Color** 중 하나로 선택하세요.



Attenuation Distance 5



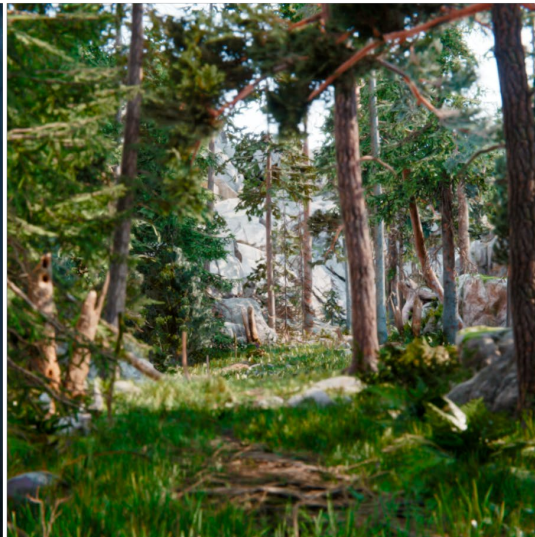
Attenuation Distance 20



Attenuation Distance 500



Fog disabled



Fog Attenuation Distance 설정은 안개가 배경으로 페이드아웃되는 방식을 조정합니다(볼류메트릭 안개 이미지).

Volumetric Fog를 활성화하면 대기 산란을 시뮬레이션할 수 있습니다. **Frame Settings**(카메라 아래 또는 HDRP Default Settings에 있음)에서 **Lighting** 아래에 있는 **Fog**와 **Volumetrics**를 선택하고, HDRP 파이프라인 에셋에서 **Volumetric Fog**를 활성화해야 합니다.

Volumetric Fog Distance는 카메라의 근거리 클리핑 평면에서 볼류메트릭 조명 버퍼 뒤쪽까지의 미터 단위 거리를 설정합니다. 이렇게 공기 중에 떠 있는 머티리얼로 대기를 채워, 범위 내의 게임 오브젝트를 부분적으로 가리게 됩니다.



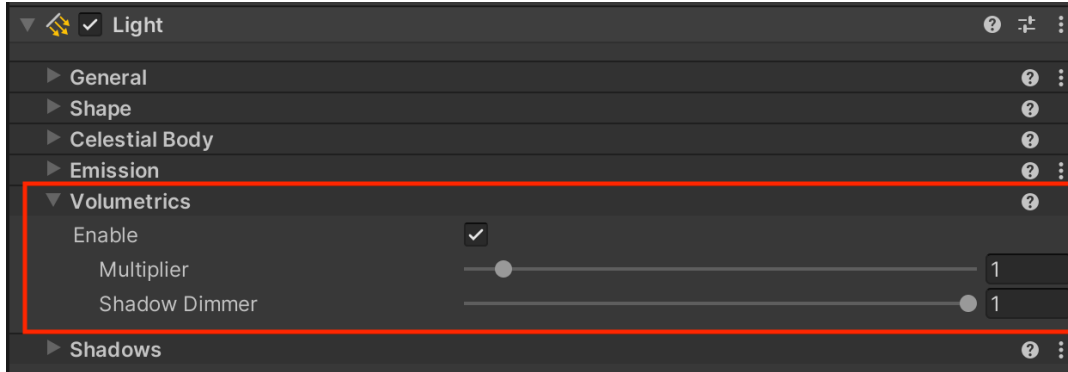
볼류메트릭 안개는 전경 지오메트리를 정확하게 표현합니다.

볼류메트릭 조명

볼류메트릭 조명을 사용하면 석양이 질 때 구름 뒤로 비치는 **тус새 빛살**이나 나뭇잎 사이로 쏟아지는 빛 등 극적인 햇빛의 렌더링을 시뮬레이션할 수 있습니다.

면 광원을 제외한 모든 조명의 Light 컴포넌트에는 **Volumetrics** 그룹이 있습니다. **Enable**을 선택한 후, **Multiplier**와 **Shadow Dimmer**를 설정합니다. **Real-time**이나 **Mixed Mode** 광원은 볼류메트릭 안개 내에서 '갓 레이(God Ray)' 효과를 생성합니다.

Multiplier는 강도를 조정하고, Shadow Dimmer는 그림자가 드리우는 표면에 의해 빛이 줄어드는 방식을 제어합니다.



Light 컴포넌트의 Volumetrics 섹션

빛줄기는 볼류메트릭 안개 내에서만 표시되므로 안개의 Base Height와 Maximum Height를 조정하여 감쇠를 제어할 수 있습니다.

i 볼류메트릭 조명 및 그림자 팁

볼류메트릭 조명과 그림자로 극적인 느낌을 더하고 싶으신가요? 다음 팁을 참고해 보세요.

- 건물이나 나무처럼 광원을 가리는 오브젝트는 그림자를 드리우며, 해당 오브젝트의 가장자리로 갈수록 빛줄기가 더욱 선명해집니다.
- 빛줄기의 모습은 카메라의 위치와 광원에 대한 상대 각도에 따라 달라집니다. 카메라 방향과 빛의 방향이 일치할수록 광선이 더 선명해집니다.
- 광원의 밝기와 색상은 갓 레이(God Ray)의 모습에 영향을 줍니다. 강도가 높을수록, 그리고 특정 색상에 따라 광선이 더 선명해집니다.
- 씬의 볼류메트릭 안개, 먼지, 기타 파티클은 빛을 산란시켜 광선을 더 선명하게 만듭니다. VFX 그래프로 빛줄기에 디테일과 애니메이션을 추가할 수 있습니다.



방향 광원의 볼류메트릭 옵션을 활성화하여 볼류메트릭 안개 사이로 빛줄기를 표시합니다.



아래 예시에서는 방향 광원과 스포트 광원에 적용된 Volumetric Multiplier의 효과를 확인할 수 있습니다.

Volumetric Multiplier 1



Volumetric Multiplier 2



Volumetric Multiplier 4

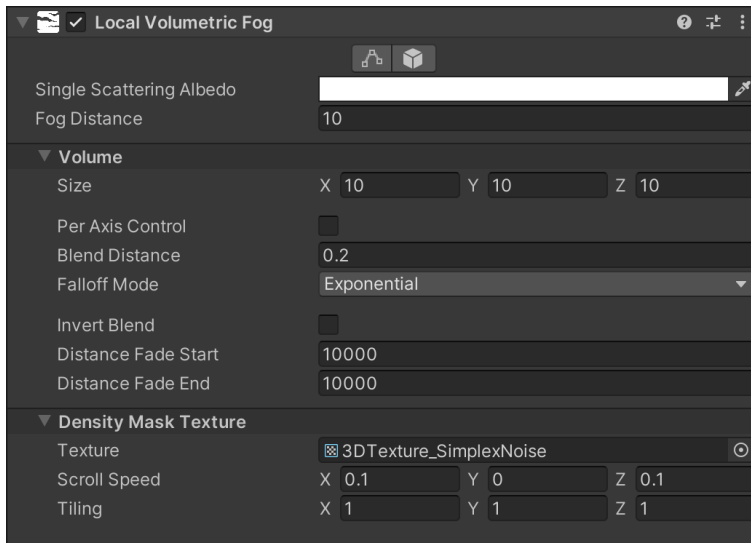


HDRP 샘플에서 천장 스포트라이트에 적용된 볼류메트릭 조명

Local Volumetric Fog

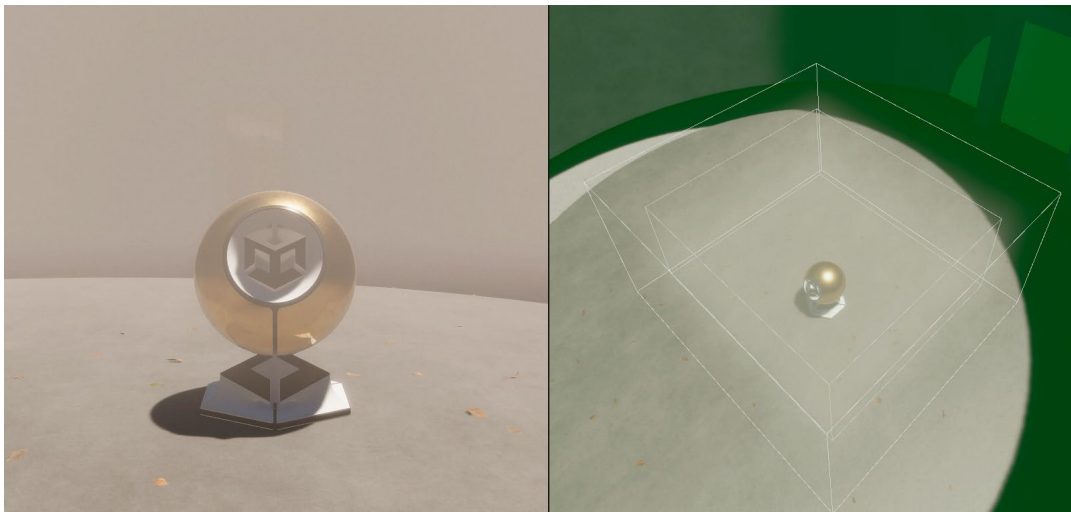
Fog 오버라이드보다 더 세부적으로 안개 효과를 조정하고 싶은 경우, HDRP에서 추가로 제공하는 **Local Volumetric Fog** 컴포넌트(이전 버전의 HDRP에서는 Density Volume)를 사용하면 됩니다.

Local Volumetric Fog는 볼륨 시스템 외부에 있는 별도의 컴포넌트입니다. 메뉴(**GameObject > Rendering > Local Volumetric Fog**)에서 **Local Volumetric Fog** 게임 오브젝트를 생성하거나 계층 구조 창에서 오른쪽 클릭하고 **Rendering > Local Volumetric Fog**를 선택합니다.



Local Volumetric Fog 컴포넌트

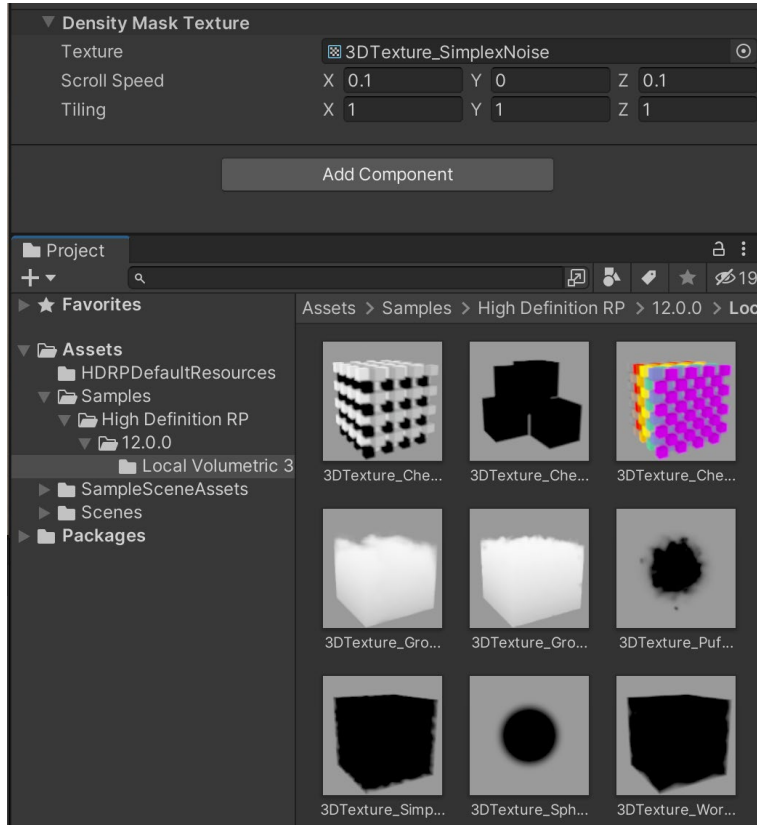
그러면 안개로 채워진 바운딩 박스가 생성됩니다. 크기, 축 제어, 블렌딩/페이드 옵션을 조정하세요.



바운딩 박스에 로컬 볼류메트릭 안개가 표시됩니다.

기본적으로 안개는 균일하게 생성되지만, **Density Mask Texture** 하위 섹션에 있는 **Texture** 필드에 3D 텍스처를 적용할 수 있습니다. 3D 텍스처를 사용해 안개 모양을 더 유연하게 제어할 수 있습니다. 패키지 관리자의 **Local Volumetric 3D Texture Samples**에서 예시를 다운로드하거나 [기술 자료에 소개된 과정](#)을 따라 직접 Density Mask를 만들어 보세요.

애니메이션 효과를 주기 위해 **Scroll Speed** 값을 추가하고 **Tiling**을 조정하면, 볼류메트릭 안개가 씬에서 자연스럽게 피어오르게 만들 수 있습니다.



Local Volumetric 3D Texture Samples의 Density Mask Texture 예셋

참고: HDRP는 성능을 개선하기 위해 Local Volumetric Fog를 복셀화(voxelize)하는데, 이 경우 안개가 매우 거칠게 보일 수 있습니다. 앨리어싱을 줄이려면 Density Mask Texture를 사용하고 Blend Distance 값을 늘려 안개의 가장자리를 부드럽게 조정하세요. HDRP 파이프라인 예셋에서 로컬 볼류메트릭 해상도를 256x256x256까지 높여 더 정확한 대규모 효과를 구현할 수 있습니다.

HDRP 물 시스템

물은 세상 어디에나 존재합니다. 이제 Unity의 물 시스템으로 그 어느 때보다 손쉽게 게임 월드에 물을 구현할 수 있습니다. 프로젝트에 외딴 열대 석호나 얼음으로 뒤덮인 험한 피오르드를 디자인하고 계신가요? 클릭 몇 번만으로 HDRP 환경에 물 요소를 추가할 수 있습니다.



물 시스템으로 바다, 강물, 호수를 추가할 수 있습니다.

물 시스템의 몇 가지 주요 기능은 다음과 같습니다.

- **물리 기반 물 렌더링:** 물 시스템에는 평활도, 굴절, 빛 산란을 조정할 수 있는 프로퍼티가 포함된 물리 기반 물 셰이더가 있습니다.
- **물 움직임:** 물 시스템은 너울, 물결, 잔물결, 흐름 등을 포함한 수면 변형을 시뮬레이션합니다.
- **수중 효과:** 카메라가 수면 아래에 잠기면 수중 렌더링이 물의 물리적 특성을 바탕으로 빛의 굴절과 흡수를 재현합니다.
- **Foam:** 거품은 수면 시뮬레이션과 풍속을 바탕으로 자동으로 형성될 수 있습니다. 또한 로컬 거품 생성기가 배가 지나간 자리와 같은 작은 영역에 하얀 물 효과를 시뮬레이션하여 거품을 표현할 수 있습니다.
- **통합:** 마스크와 데칼, 디포머와 같은 아티스트 친화적인 컴포넌트로 수면을 주변 프랍과 터레인에 연결할 수 있습니다.
- **성능:** 물 시스템은 다양한 플랫폼에 최적화되어 있으며 시각화를 위한 다양한 디버그 모드를 제공합니다.



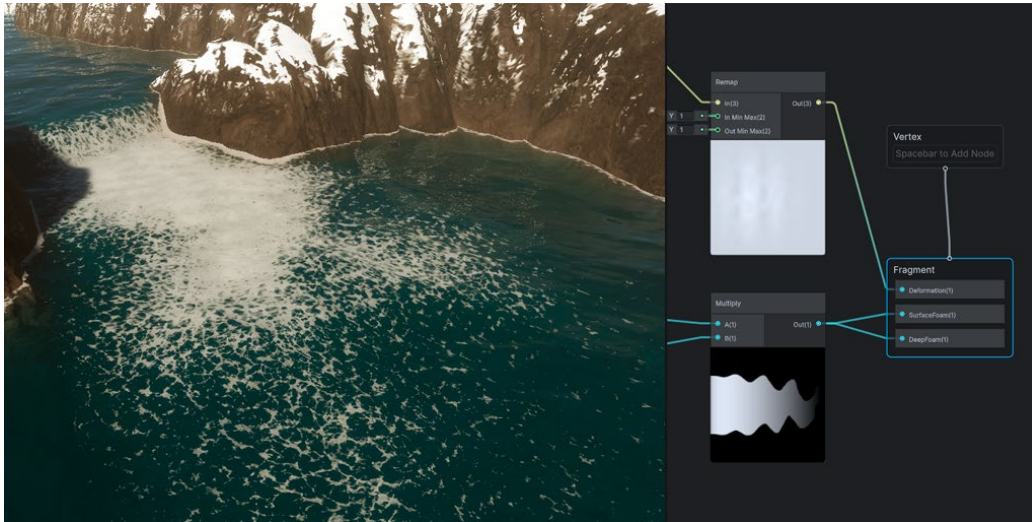
물 시스템은 유연하고 커스터마이징이 가능합니다.

Unity 6.1의 새로운 기능

새로운 물 시스템 기능이 포함된 Unity 6.1(HDRP 17)로 시작할 것을 권장합니다.

- **수면 트랜스폼 지원 개선:** 이제 Infinite Oceans 표면과 Instanced Quad 표면이 이동, 회전, 음수 스케일링을 지원하며 표면 반전이 가능합니다.
- **물 데칼 ShaderGraph 타겟:** 새로운 ShaderGraph 타겟을 사용하면 아틀라스에 물 변형과 거품을 직접 출력할 수 있습니다. 이에 따라 Custom Render Texture를 사용할 필요가 없어집니다. 거품 애니메이션이 이제 물의 방향을 따르게 되어 더 동적인 효과를 부여합니다.
- **향상된 수중 렌더링:** 이제 볼류메트릭 안개가 갓 레이(god ray)와 빔줄기를 지원하여 수중 환경의 몰입감을 한층 높여 줍니다.

- **Shader Graph의 새로운 Camera Height 노드:** 새로운 Shader Graph 노드를 사용하면 수면을 기준으로 카메라의 높이를 샘플링할 수 있습니다. 이에 따라 카메라 렌즈에 맺힌 물방울과 같은 효과를 구현할 수 있습니다(아래의 **Underwater** 샘플 씬 참조).



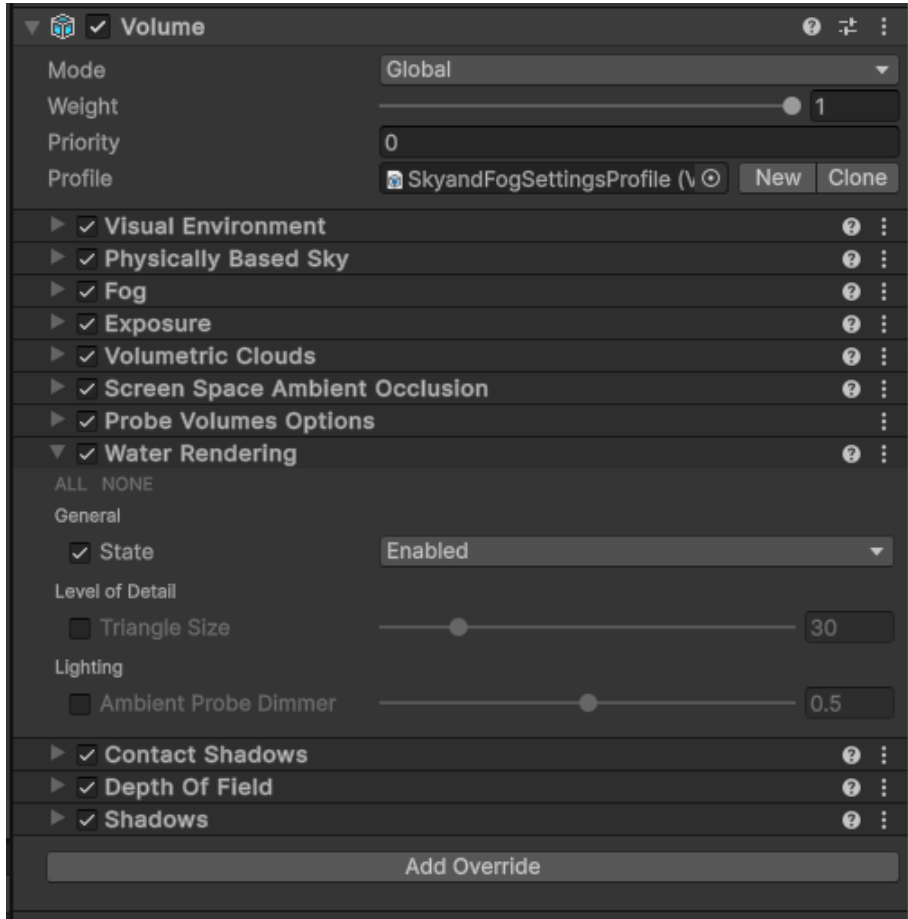
새로운 ShaderGraph 타겟이 거품 출력을 지원합니다.

Cave 및 **RollingWave** 샘플 씬을 살펴보려면 Unity 6.1 이상이 필요합니다.

물 시스템 시작하기

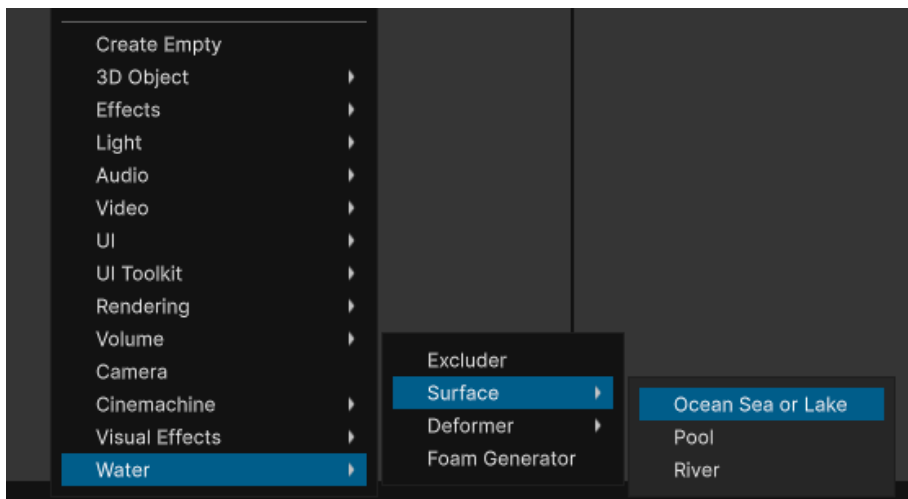
프로젝트에 물 시스템을 적용하는 방법은 다음과 같습니다.

- 물이 필요한 품질 수준에 따라 각 렌더 파이프라인 에셋에서 물을 활성화하고 설정합니다.
- 카메라의 프레임 설정에서 물을 활성화합니다(**Edit > Project Settings > Graphics > HDRP Global Settings**).
- 씬에서 **Water Rendering** 볼륨 오버라이드를 사용하여 카메라의 위치를 바탕으로 어디에서 물 렌더링이 활성화될지 제어합니다.



볼륨 오버라이드에서 Water Rendering을 활성화합니다.

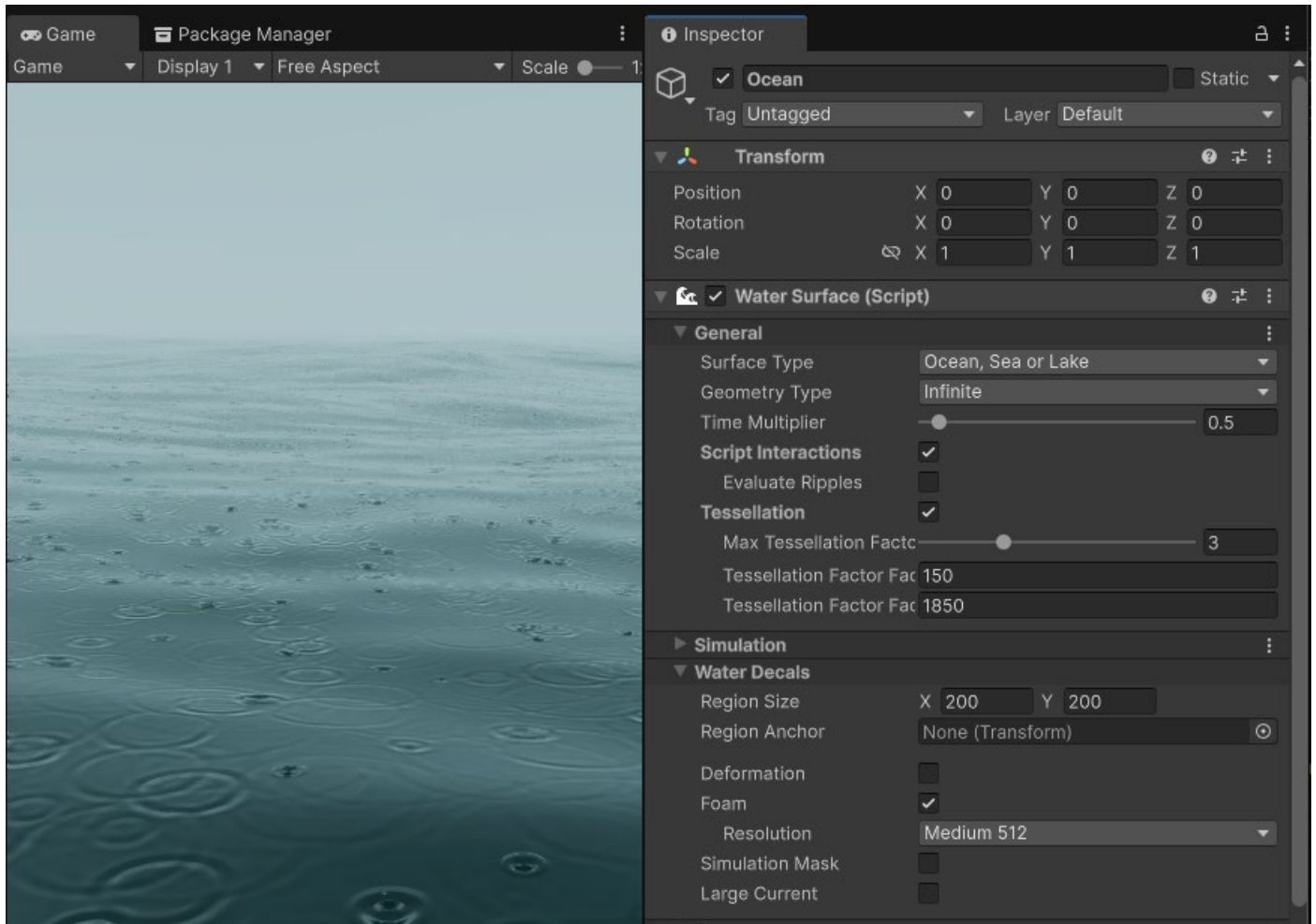
물을 활성화한 다음 GameObject 메뉴에서 사전 설정된 수역을 추가할 수 있습니다. HDRP는 Pool, River, Ocean 세 가지의 수면 유형을 제공합니다.



GameObject 메뉴에서 물 오브젝트를 추가합니다.

Water Surface 컴포넌트

수면 오브젝트에는 물의 일반 파라미터를 제어하는 **Water Surface** 컴포넌트 스크립트가 있습니다.



Water Surface 컴포넌트

수면의 시뮬레이션과 렌더링을 설정하는 파라미터는 다음과 같습니다.

- **General:** 수역의 전체 유형(예: Ocean/Sea/Lake, River, Pool)과 수면 렌더링에 사용할 지오메트리(예: Quad, Custom, InstancedQuads, Infinite)를 정의합니다.
- **Simulation:** 바람과 달이 수면에 영향을 주는 방식을 모방하여 파도 패턴과 잔물결이 만들어지는 방식을 제어합니다.
- **Water Decals:** 물 데칼의 중심과 크기를 설정합니다(아래 참조). 이 설정을 사용하여 변형, 거품, 시뮬레이션 마스크, 물 흐름을 토글하고 텍스처 해상도를 설정합니다.



- **Appearance:** 물의 색상, 평활도, 굴절, 빛 산란을 결정합니다. 커스틱과 특수한 수중 설정으로 물리 기반 셰이딩을 향상합니다.
- **거품:** 파도의 꼭대기 부분, 수중 오브젝트의 주변, 해안선을 따라 보이는 거품의 모습과 동작을 제어합니다.
- **Miscellaneous:** 렌더링 레이어 마스크와 디버그 모드를 제어합니다.

물 데칼(Unity 6.1)

Unity 6.1에서는 물 데칼을 사용하여 물 효과를 세밀하게 제어할 수 있습니다. 물 데칼은 월드 공간에 적용된 [Shader Graph 마스터 스택](#)입니다. 몇 가지 응용법은 다음과 같습니다.

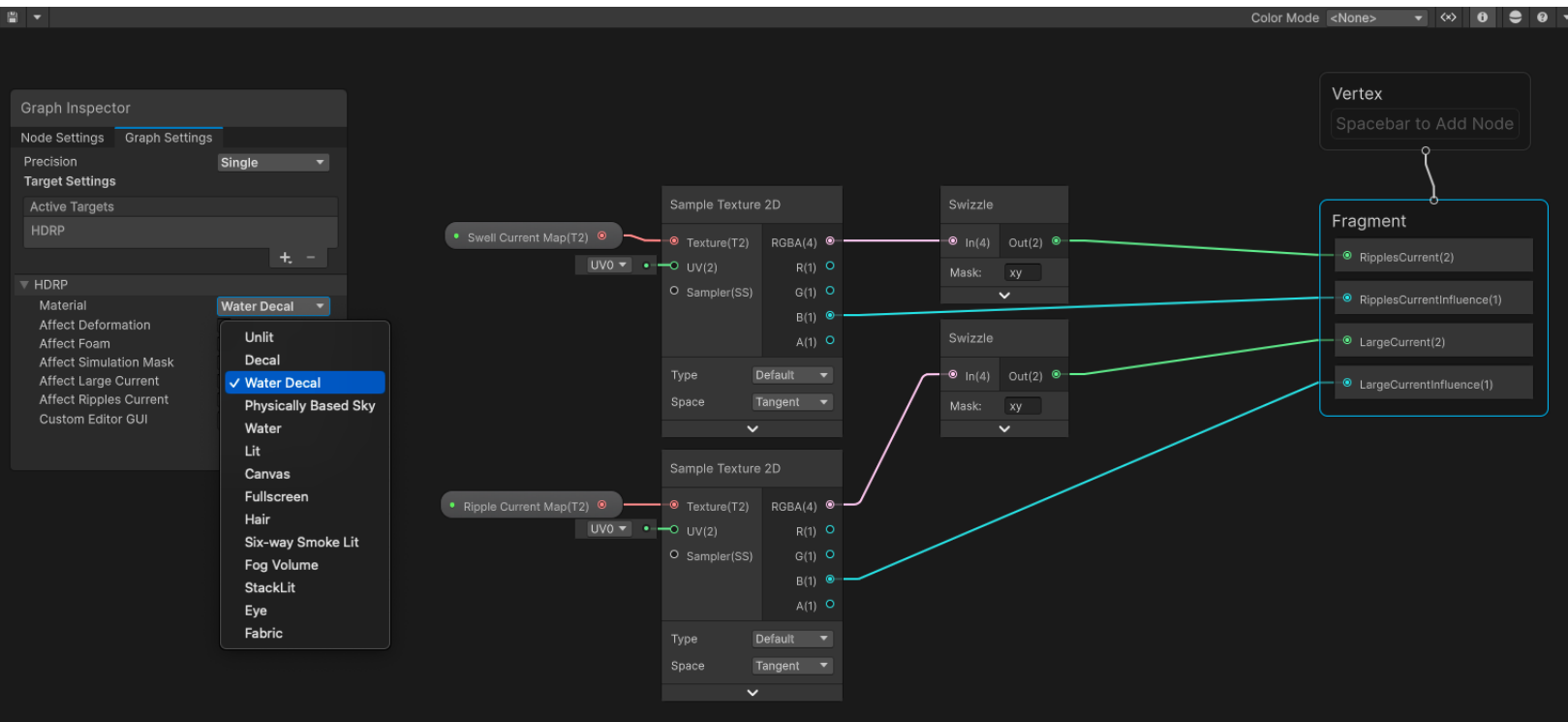
- **Currents:** 데칼이 물의 국지화된 흐름을 정의할 수 있습니다.
- **Deformation:** 수면의 높이와 변위를 수정합니다.
- **Foam:** 물이 오브젝트 또는 터레인과 상호 작용할 때 화이트워터 효과를 추가합니다.

물 데칼을 만들려면 **Create** 또는 **GameObject** 메뉴로 이동하여 **Water > Water Decal**을 선택합니다. 그러면 씬에 게임 오브젝트가 추가됩니다. 효과를 적용하려는 영역으로 데칼을 옮기세요.

Water Surface 컴포넌트의 **Water Decal** 프로퍼티 아래에 있는 설정을 사용하여 텍스처 해상도를 수정하거나 특정 옵션을 토글합니다.

물 데칼의 Shader Graph를 편집하여 데칼이 물에 영향을 주는 방식을 구성합니다. [마스크와 현재 물 데칼을 활성화](#)한 후에는 그래프 인스펙터를 통해 다음과 같은 물 기능을 추가할 수 있습니다.

- HorizontalDeformation
- SimulationMask
- SimulationFoamMask
- LargeCurrent
- LargeCurrentInfluence
- RipplesCurrent
- RipplesCurrentInfluence



물 데칼 효과는 Shader Graph 마스터 스택을 사용합니다.

물 데칼 영역은 기본적으로 카메라에 고정되어 있으며, 원하는 경우 게임 오브젝트에 고정할 수도 있습니다. 물 데칼을 사용하는 방법을 자세히 알아보려면 [Water 패키지 샘플](#)의 **RollingWave** 씬을 참조하세요.

Unity 6의 Water 샘플

HDRP 물 시스템에 익숙해지는 가장 좋은 방법은 다양한 사용 사례를 보여 주는 샘플 씬을 살펴보는 것입니다. 패키지 관리자에서 **High Definition Render Pipeline(HDRP)** 패키지를 선택하고 **Samples** 탭에서 **Water Samples**를 설치합니다.

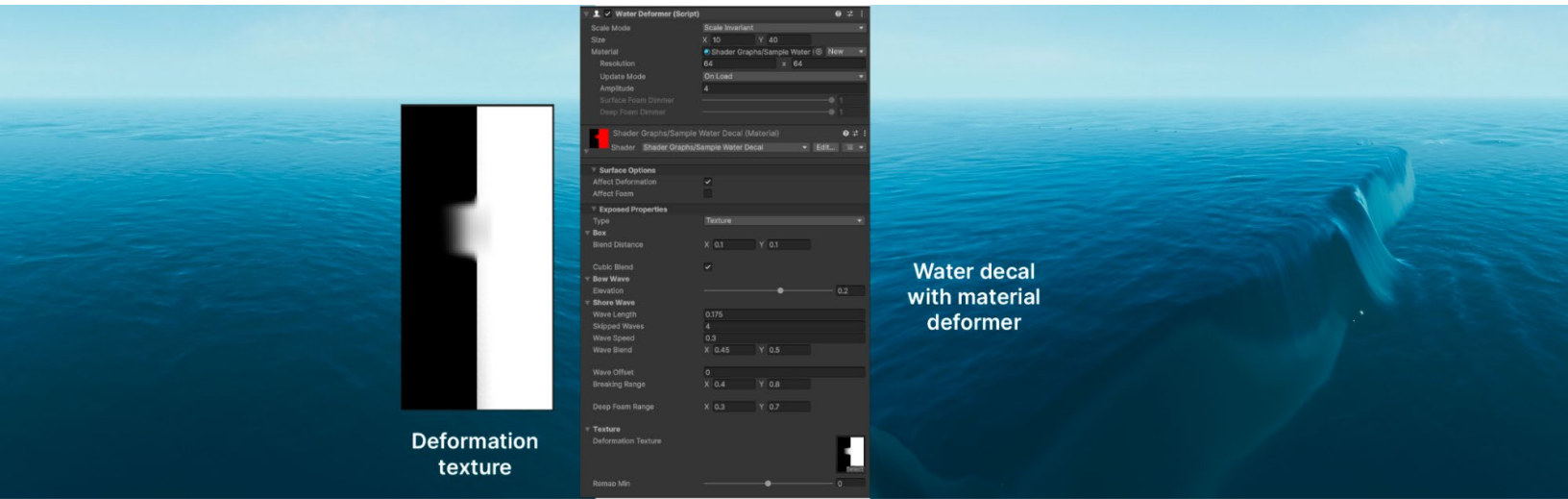
Unity 6.1에 맞게 업데이트된 이 샘플은 프로젝트에 사실적인 물을 적용하기 위한 시작점이 되어 줍니다.

빙하

빙하 샘플은 강에서 바다와 같은 더 큰 수역으로 전환하는 방법을 보여 줍니다. Unity 6.1에 맞게 업데이트된 빙하 샘플은 이제 **Mask and Current Water Decal** 워크플로(**Project Settings > Graphics**에서 활성화)를 사용합니다.

물 데칼은 기존의 물 디포머 시스템을 대체하며 이제 모든 물 변형을 처리합니다.

머터리얼 디포머가 적용된 물 데칼은 중앙 폭포의 형태를 만들며, 변형 텍스처를 기반으로 물의 높이를 상승시킵니다. Custom Render Texture를 사용하는 [Decal Projector 컴포넌트](#)는 떨어지는 물을 시뮬레이션하는 애니메이션화된 텍스처를 생성합니다.



물 데칼은 수면을 변형합니다.

Water Simulation 데칼은 강 구간에서 바다의 너울이 발생하지 않도록 하고, 강의 시작 지점에서 거품을 생성합니다. **Water Current 데칼**은 물의 흐름을 강에서 바다로 유도합니다.



빙하 샘물은 강을 바다에 연결합니다.

픽셀 뎀스 값을 비교하여 강기슭을 따라 거품이 추가됩니다. 강둑과 수면의 차이가 작으면 거품이 적용됩니다.



강 가장자리에 거품이 나타납니다.

커스틱 효과는 빛의 굴절을 시뮬레이션하여 물 상호 작용의 사실감을 높입니다. 얼음 벽의 커스틱은 Decal Projector 컴포넌트를 사용하여 투사되며, GetCausticsBuffer 함수를 통해 실시간으로 업데이트됩니다.

강둑의 경우, Water Surface 컴포넌트에서 곧바로 커스틱이 활성화됩니다. 단, 물 흐름 맵의 영향은 받지 않기 때문에 물의 움직임과 관계없이 조명 효과가 일관되게 구현됩니다.



커스틱 및 얼음 조각

부유하는 얼음 조각은 수면 위의 부력을 시뮬레이션하는 FloatingIceberg 스크립트를 사용하여 물의 흐름과 함께 움직입니다. 각 얼음 조각은 Foam Generator를 포함하므로, 떠다니는 동안 조각 뒤쪽으로 거품 효과가 생성됩니다.

섬

이 씬은 해안과의 상호 작용 및 부유하는 오브젝트가 있는 개방 수역을 보여 줍니다. 하나의 물 데칼은 다양한 용도를 갖습니다.

- 데칼 마스크는 **Affect Simulation Mask** 옵션을 사용하여 너울과 물결의 일부를 마스킹합니다.
- 이 시뮬레이션 마스크 텍스처는 Water Decal Shader Graph에서 거품을 마스킹하는 데도 사용됩니다.
- Current Map 텍스처가 **Affect Large Current** 옵션을 사용하여 수면의 흐름 방향을 변경합니다.



섬 샘플에서는 부서지는 해안 파도를 볼 수 있습니다.

해안 근처의 사실감을 높이기 위해, 해안 파도는 해안선을 따라 부서지는 모습을 시뮬레이션합니다. 3인치 캡슐 컨트롤러가 물속을 이동하면서 수면을 교란하는 동적 항적을 생성합니다. 한편 **Water Excluder** 게임 오브젝트는 씬에 있는 배처럼 부유하는 오브젝트 내에 물이 나타나지 않도록 합니다.



3인치 캡슐 컨트롤러가 물속을 이동합니다.

풀

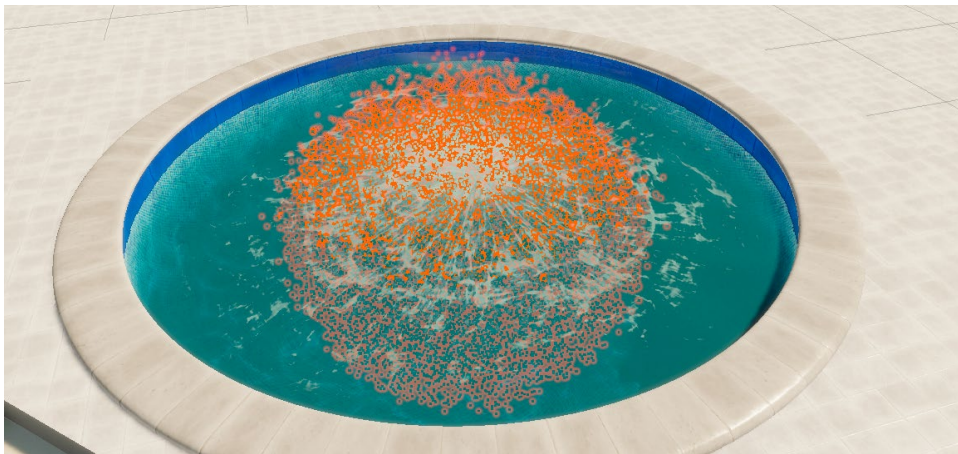
이 씬은 부력과 디테일한 반사를 보여 줍니다. 풀에 비치볼을 떨어뜨리면 각 효과가 수면을 따라 국지적인 변형과 스플래시 효과를 트리거하는 것을 볼 수 있습니다.

비치볼은 질량과 부피를 고려한 커스텀 부력을 사용하여 수면 위를 떠다닙니다.



Pool 샘플에서 부력이 있는 오브젝트들이 떠다니고 있습니다.

물의 다양한 반사를 보여 주기 위해 HDRP 에셋에서 **Screen Space Reflection on Transparent**가 활성화되어 있습니다. 이에 더해 Caustics 설정의 Directional Shadow가 풀 수중의 그늘진 영역에서 커스틱 효과를 어둡게 만드는 것을 볼 수 있습니다.



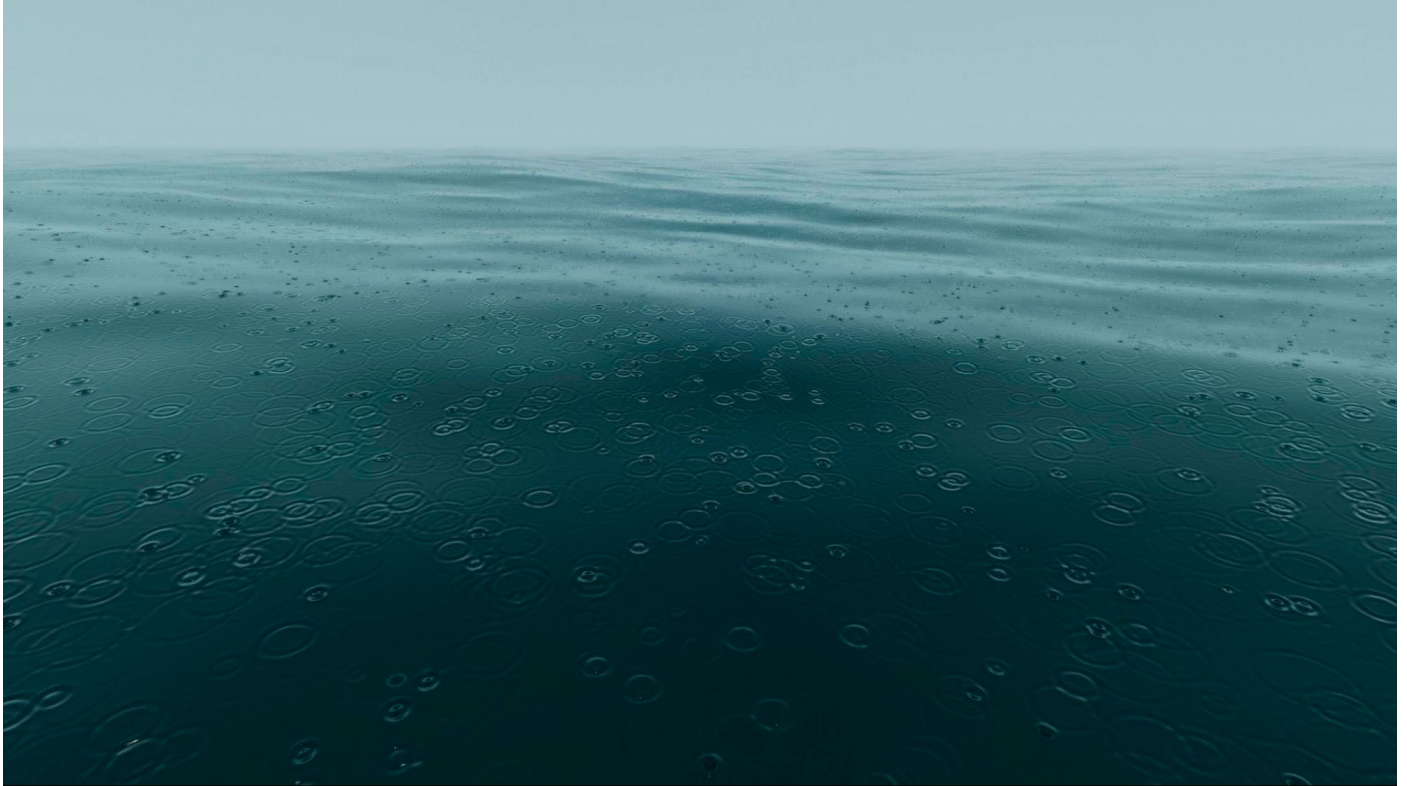
VFX Graph와 데칼을 통해 온수 욕조 방울이 생성됩니다.

두 번째 온수 욕조 시뮬레이션에는 Bubbles 게임 오브젝트가 있습니다. 이 오브젝트는 Decal Projector 및 연결된 VFX Graph를 사용하여 움직이는 물 효과를 생성합니다.

비

이 씬은 바다 표면과 상호 작용하는 비를 시뮬레이션합니다. Custom Render Texture가 실시간으로 빗방울의 잔물결을 생성하며, Shader Graph 머티리얼이 수면에 동적으로 잔물결을 적용합니다.

조정 가능한 설정을 사용하여 빗방울의 속도, 방향, 강도를 제어하세요.



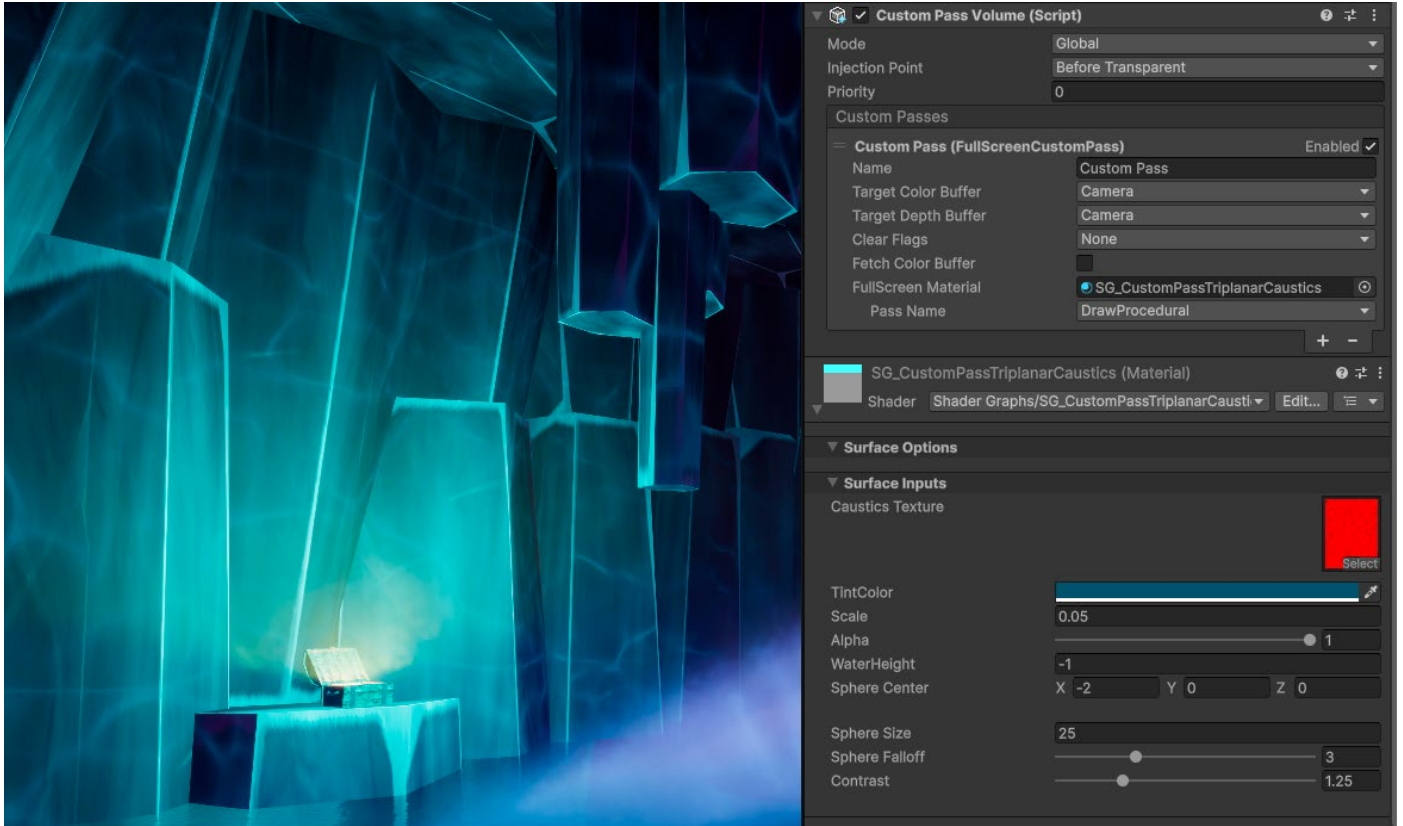
Custom Render Texture를 통해 빗방울의 잔물결을 생성할 수 있습니다.

이어지는 샘플에서는 6.1 이상 버전의 새로운 워크플로를 소개하고 있으므로 Unity 6.1 이상이 필요합니다.

동굴

이 씬에서는 Unity 6.1 이상에서 커스틱 텍스처의 다양한 용도를 살펴봅니다. 여기서는 동굴 환경에서 볼 수 있는 조명 효과를 적용합니다. 이렇게 하면 물을 통과하며 굴절되고 모아진 빛이 주변 표면에 일렁이는 패턴을 만들어 내는 현상을 시뮬레이션할 수 있습니다.

Custom Pass는 전체 화면 머티리얼과 Shader Graph를 사용하여 동굴의 벽과 천장에 커스틱 조명 효과를 투사합니다. 셰이더는 수면에서 커스틱 텍스처를 샘플링합니다. **SG_CustomPassTriplanarCaustics** 머티리얼 설정을 조정하여 모양을 미세 조정할 수 있습니다.



Custom Pass는 커스틱 조명 효과를 투사합니다.

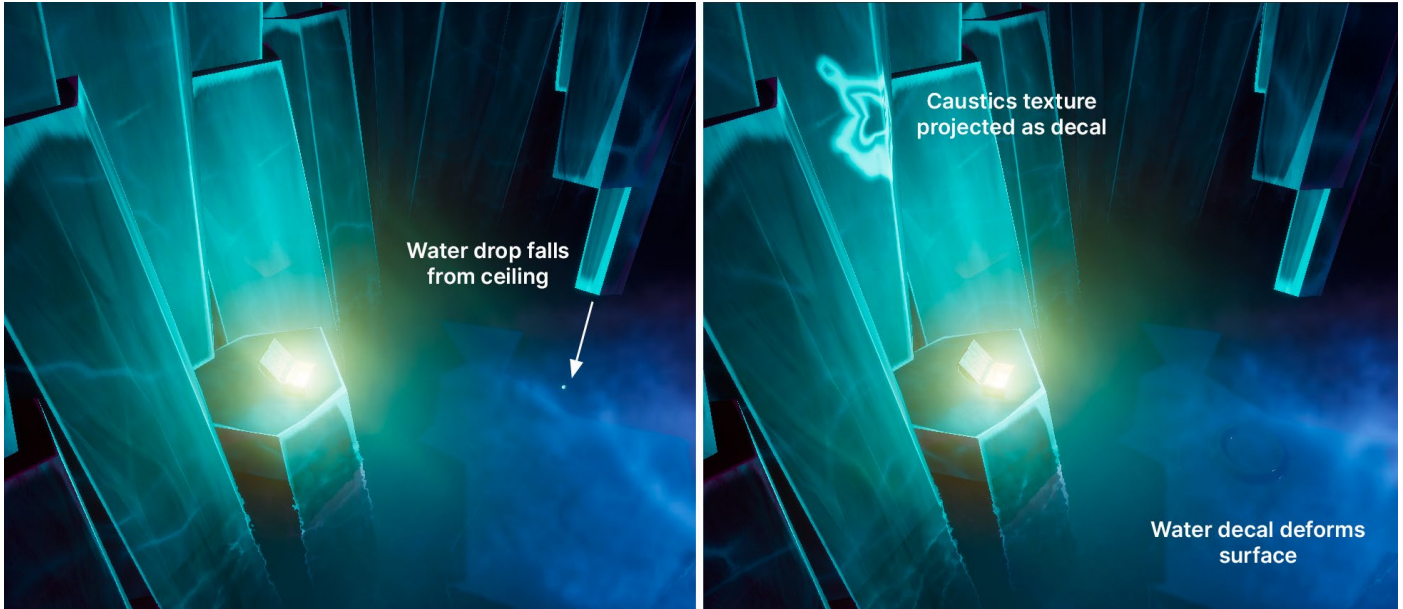
Shader Graph를 편집하여 각 축을 따라 커스틱 텍스처가 어떻게 투사되는지 살펴봅니다. 셰이더는 위를 향하는 평평한 표면에 커스틱이 나타나지 않도록 거리와 -Y 방향을 따라 효과를 감쇠시킵니다. 또한 셰이더는 물 시스템이 커스틱을 처리하는 수면 아래에 커스틱이 나타나는 것도 방지합니다.



동굴에서는 커스틱 투사와 볼류메트릭 안개를 볼 수 있습니다.

빛 기둥의 경우, 커스텀 머티리얼을 갖는 **Local Volumetric Fog**를 사용하여 수면에서 반사되는 빛 패턴을 시뮬레이션합니다. Shader Graph가 Water Surface 컴포넌트에서 커스틱 텍스처를 샘플링하여 부드러운 대기 효과를 생성합니다.

마지막으로, 천장에서 애니메이션화된 물방울이 떨어지며 수면 위에 잔물결을 일으킵니다. Decal Projector가 인근 벽에 나타나는 커스틱 텍스처를 시뮬레이션합니다.



물방울이 표면을 변형하고 벽에 커스틱을 투사합니다.

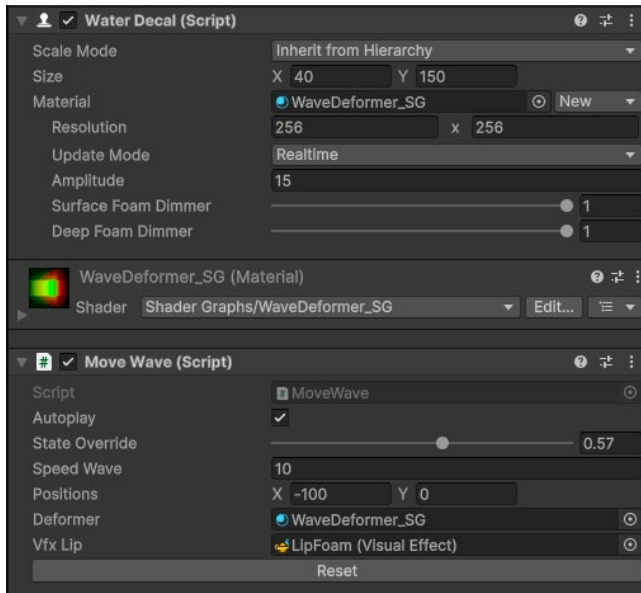
밀려오는 파도

이 씬에서는 Unity 6.1의 물 데칼 변형을 통해 변형 텍스처를 사용하여 크게 밀려오는 파도를 확인할 수 있습니다. Shader Graph와 커스텀 이동 스크립트를 사용하여 파도를 생성합니다.



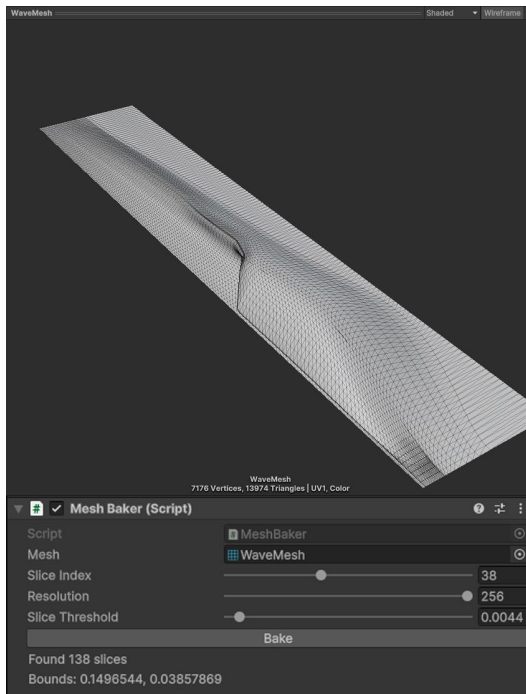
변형 텍스처는 파도를 생성합니다.

파도를 생성하기 위해 Shader Graph에서 변형 텍스처가 샘플링되어 파도의 모양을 정의합니다. 셰이더가 UV의 대칭을 조정하고 블렌딩 영역을 사용하여 파도를 주변 물에 부드럽게 이어 줍니다. 그런 다음 MoveWave 스크립트가 +X축을 따라 파도를 이동시키며 애니메이션화합니다.



Water Decal 스크립트가 물을 변형합니다.

변형 텍스처 자체는 여러 개의 슬라이스로 구성된 사전 제작된 메시로부터 생성되며, 각각은 서로 다른 파도 상태를 나타냅니다. 이 슬라이스는 MeshBaker 스크립트가 버텍스 위치를 기록하고 텍스처에 저장하여 처리합니다. 각 텍스처 행은 하나의 슬라이스에 대응되며, 각 텍스처 채널은 X, Y, Z 변형을 인코딩합니다.



커스텀 Baker를 사용하여 변형 텍스처를 생성합니다.



스크립트를 테스트하려면 **Mesh Baker** 게임 오브젝트를 활성화하고 설정을 조정한 다음 인스펙터에서 **Bake**를 누릅니다.

수중

Underwater 샘플에서는 카메라가 수중으로 들어가면 어떤 일이 일어나는지 볼 수 있습니다. 여기서는 물과 공기가 맞닿는 지점이 물 경계선으로 구분됩니다. 물 경계선은 기본적으로 수면 위의 뷰와 수중 뷰를 구분하는 뚜렷한 경계입니다. 더 자연스러운 느낌을 부여하려면 **Custom Pass**를 사용하여 전환에 더 부드러운 효과를 줄 수 있습니다.

이 효과는 물 경계선 주변에 블러를 적용하는 **Full Screen Shader Graph**를 사용합니다. 셰이더는 특수 수중 버퍼를 샘플링하여 어느 픽셀이 수중에 있는지 감지합니다. 그런 다음 **HD Scene Color** 노드를 사용하여 물 경계선의 가장자리만 블러 처리해서 전환에 더 부드러운 효과를 줍니다.



셰이더는 물 경계선을 블러 처리합니다.



셰이더는 더 사실적인 효과를 위해 물 경계선 위의 이미지를 약간 늘려 메니스커스 효과를 구현합니다.



Custom Pass는 화면 위에 물방울을 더합니다.

Custom Pass는 화면 UV를 왜곡하는 물방울을 더한 다음 **HD Scene Color** 노드를 샘플링합니다. 그러면 **Custom Render Texture**가 각 프레임을 이동시켜 드리핑 효과를 생성합니다.



이 씬에서는 배경에 생동감을 부여하는 VFX Graph 효과도 볼 수 있습니다.

- 식물은 버텍스 색상과 사인파를 사용하여 애니메이션화됩니다.
- 물 볼륨 안에서 물고기 떼가 역동적으로 헤엄칩니다.
- 물방울은 **Sample Water Surface** 노드를 사용하여 수면 위에 나타나지 않도록 처리됩니다.
- 떠다니는 파티클은 탁한 수중 환경을 시뮬레이션합니다.



볼류메트릭 그림자가 수중 환경에 사실감을 더합니다.

볼류메트릭 안개는 물속에서 빛이 산란되는 방식을 시뮬레이션합니다. Water Surface 컴포넌트에서 커스텀 텍스처를 샘플링하여 빛줄기가 물에 침투하는 효과를 생성함으로써 극적인 볼류메트릭 그림자를 더합니다.

셰이더 및 머티리얼

머티리얼은 렌더링 과정에서 오브젝트가 빛을 어떻게 반사하거나 발산하는지 결정하는 중요한 역할을 합니다. 머티리얼은 오브젝트에 금속, 유리, 나무뿐 아니라 추상적이거나 마법 같은 느낌을 부여할 수 있습니다.

머티리얼에는 셰이더 오브젝트에 대한 레퍼런스가 있습니다. 해당 셰이더 오브젝트가 **머티리얼 프로퍼티**를 정의하는 경우, 머티리얼에도 색상이나 텍스처 레퍼런스와 같은 데이터가 저장됩니다.

셰이더는 GPU에서 실행되어 각 픽셀의 색상을 결정하는 프로그램으로, 텍스처, 조명 정보, 머티리얼 프로퍼티 등이 담긴 입력 데이터를 바탕으로 계산을 수행합니다.

머티리얼 샘플



HDRP에는 패키지 관리자에서 다운로드할 수 있는 머티리얼 샘플이 있습니다.

HDRP 머티리얼 샘플에는 HDRP 전용 머티리얼 및 셰이더 기능에 대한 다양한 예시가 포함되어 있습니다. 샘플에서 피하 산란, 변위, 이방성 등의 효과를 살펴볼 수 있습니다. 포함된 Shader Graph는 [Lit Shader](#) 스택, [Fabric 마스터 스택](#), [Decal 마스터 스택](#) 등의 [마스터 스택](#)(아래 내용 참고)을 사용합니다.

Fabric, Hair, Eye 마스터 노드의 경우 일반적으로 Shader Graph에서 아티스트의 디테일 작업이 이루어져야 합니다. 이러한 샘플은 사용자가 직접 셰이더를 만들기 위한 좋은 시작점이 됩니다.



Fabric 샘플

Eye 샘플은 정교하게 설계된 메시와 특수 UV 설정 및 특정 임포트 스케일 인자를 사용합니다. 이와 비슷한 품질의 눈을 만들려면 3D 모델링 소프트웨어에서 눈 메시를 살펴보고 구성과 UV 레이아웃에 대해 알아보세요.

일부 머티리얼을 올바르게 시각화하려면 **레이트레이싱**을 지원하는 GPU가 필요합니다.

머티리얼 배리언트

Unity의 복잡한 머티리얼 라이브러리를 효율적으로 관리하고 변경 사항을 적용할 방법이 필요한 경우, 머티리얼 배리언트를 사용하면 일반적인 머티리얼을 기반으로 하는 템플릿이나 머티리얼 프리팹을 만들 수 있습니다.



머티리얼 배리언트 예시

배리언트는 템플릿 머티리얼과 동일한 프로퍼티를 공유하고 필요한 것만 오버라이드할 수 있습니다. 게임 오브젝트를 프리팹으로 만드는 것처럼, 머티리얼 배리언트에서 특정 프로퍼티가 오버라이드되지 않도록 잠글 수도 있습니다.

예를 들어 프로젝트에 사용할 다양한 목재 머티리얼을 설정하려는 경우, Wood Base 머티리얼로 시작한 다음 Oak나 Cherry처럼 텍스처가 다른 배리에이션을 생성할 수 있습니다. 결과적으로는 Base Color 톤트 등의 일부 프로퍼티를 오버라이드하여 White Oak나 Red Oak 같은 별도의 머티리얼 배리언트를 만드는 것입니다.

머티리얼 배리언트를 사용하면 여러 프로젝트에 사용할 여러 기본 머티리얼을 갖춘 다용도 라이브러리를 구축하고, 필요에 따라 오버라이드할 수 있습니다. 따라서 아티스트가 미세 조정을 위해 머티리얼을 복제하지 않아도 되고, 특히 대형 프로젝트에서 시각적 버그나 성능 저하를 방지할 수 있습니다.

머티리얼 배리언트에 대해 자세히 알아보려면 [이 기술 자료 페이지](#)를 참고하세요.

머티리얼 프로퍼티

씬의 머티리얼에는 대부분 **HDRP Lit Shader**가 사용됩니다. 다음은 HDRP Lit Shader의 주요 프로퍼티 목록이며, 다른 HDRP 셰이더에도 이 프로퍼티가 있을 수 있습니다.

Base Color: 머티리얼의 색상과 불투명도를 제어합니다.

Metallic: 표면의 모습이 얼마나 ‘금속 느낌’을 내는지 정합니다. 값이 클수록 메탈릭(Metallic) 광택이 강해지며, 값이 낮을수록 메탈이나 유전체와 다른 모습을 나타냅니다.

Specular: 메탈이 아닌 머티리얼의 반사도를 결정합니다. 메탈릭 머티리얼의 경우 스페큘러 하이라이트 색상은 베이스 색상에서 파생됩니다.

Smoothness: 머티리얼의 반사 선명도를 결정합니다. 평활도가 높을수록 표면이 더 반짝이고 매끄러워지며, 값이 낮을수록 광택이 없는 거친 머티리얼이 됩니다.

Anisotropy: 머티리얼의 하이라이트가 한 방향으로 길게 늘어나는 시각 효과(예: 탄소 섬유 또는 브러시 처리된 메탈)를 생성합니다.



Anisotropy는 하이라이트를 한 방향으로 길게 늘입니다.

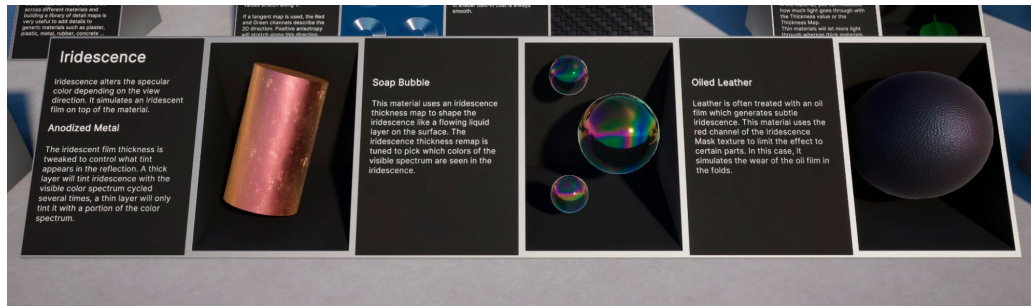
Thickness: 일반적으로 SSS(피하 산란)를 사용하여 빛이 반투명 머티리얼의 표면을 통과해서 산란되는 모습을 표현합니다.

Emission: 머티리얼이 발광할 수 있게 만듭니다. 조명의 광원으로 사용되는 오브젝트(예: 네온사인)에 유용합니다.

Normal Map: 범프나 홀 등의 표면 특성을 세부적으로 추가할 수 있는 특수한 유형의 텍스처입니다.

Detail Map: 베이스 맵에 텍스처 레이어를 추가하여 텍스처가 가까이에서 어떻게 보이는지 더 정밀하게 제어할 수 있게 합니다.

Iridescence: 비눗방울이나 특정 보석처럼 보는 각도에 따라 색상이 달라지는 것을 표현합니다.

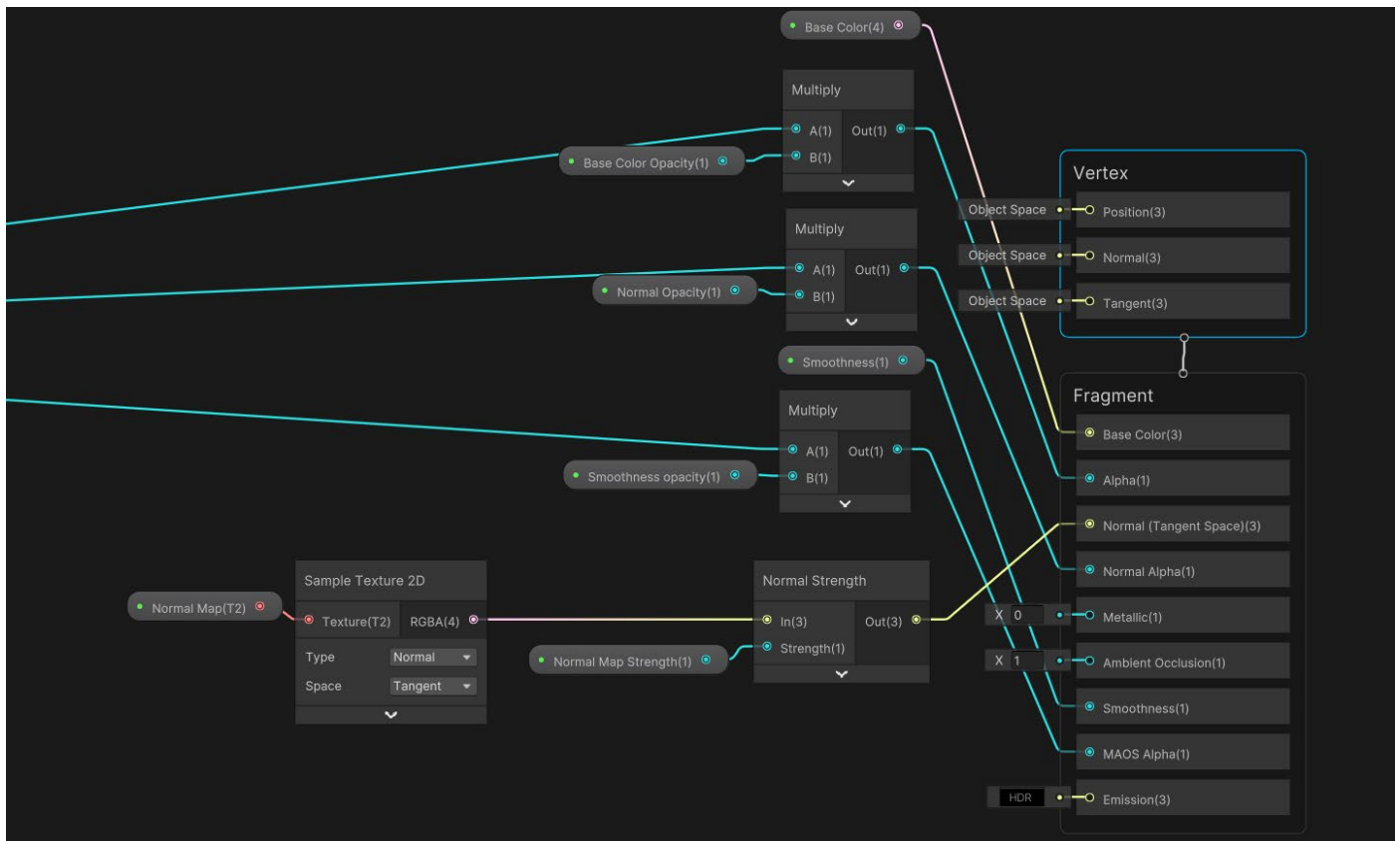


Iridescent 오브젝트는 보는 각도에 따라 색상을 바꿉니다.

Shader Graph

Shader Graph는 코드를 작성하지 않고도 시각적으로 셰이더를 제작할 수 있는 강력한 시각화 툴입니다. HLSL 같은 언어로 셰이더 코드를 작성, 디버깅하고 관리할 필요 없이 복잡한 셰이더를 제작할 수 있는 노드 기반 인터페이스를 제공합니다.

개발자가 Shader Graph 노드를 연결하여 시각적인 방식으로 복잡한 셰이더를 구성할 수 있으므로 셰이더 프로그래밍에 익숙하지 않더라도 셰이더 제작을 쉽게 시작할 수 있습니다. 셰이더 그래프의 각 노드는 수학 연산, 함수 또는 셰이더 프로퍼티를 나타냅니다.



노드 기반 셰이더 그래프를 사용하여 시각적으로 풍부한 셰이더를 제작할 수 있습니다.

셰이더 그래프는 HDRP와 완전히 통합되었으며, HDRP 전용으로 설계된 여러 노드와 기능을 제공합니다. 여기에는 HDRP의 물리 기반 조명 모델, 고급 머티리얼 유형, 포스트 프로세싱 효과에 대한 지원이 포함됩니다.

HDRP는 고해상도 비주얼을 우선시하므로, 더욱 사실적인 물리 기반 머티리얼을 제작할 수 있도록 다양한 셰이더를 지원합니다. 각 HDRP 머티리얼 유형은 셰이더 그래프의 [마스터 스택](#)에 대응됩니다.

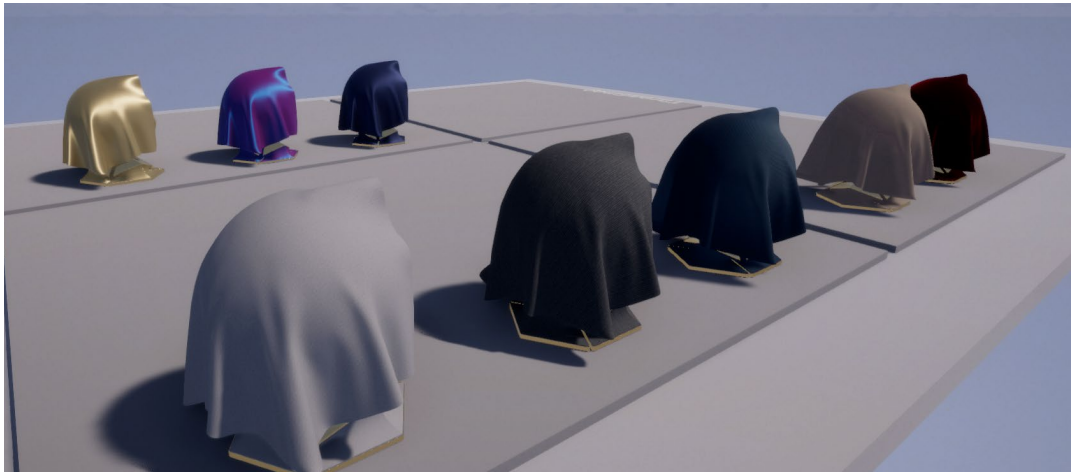
HDRP 마스터 스택

[마스터 스택](#)은 셰이더 그래프의 엔드포인트로, 셰이더의 최종 표면 형상을 정의합니다. 셰이더 그래프에 마스터 스택은 하나만 존재해야 합니다.

HDRP에는 다음과 같은 마스터 스택이 있습니다.

- **Lit Master Stack:** 물리 기반 조명으로 범용 머티리얼을 생성합니다. 사실적인 머티리얼을 위한 가장 일반적인 마스터 스택이며 피하 산란, 반투명, 이방성 등의 다양한 기능을 지원합니다.
- **StackLit Master Stack:** 레이어가 여럿인 복잡한 머티리얼을 위해 설계되었으며, 광원이 각 레이어와 상호 작용하는 방식을 세밀하게 제어할 수 있습니다. 예를 들어 StackLit Master Stack을 사용하면 베이스 레이어와 메탈릭 조각 레이어, 클리어 코트가 필요한 차량 페인트 머티리얼을 만들 수 있습니다.
- **Unlit Master Stack:** 조명과 상호 작용하지 않는 머티리얼에 적합하며, 광원과 그림자에 대한 고려 없이 색상과 텍스처를 완전히 제어합니다.
- **Decal Master Stack:** 표면에 그래픽 오버레이를 추가합니다. 최종 결과물로 작용하는 셰이더로, 표면에 먼지나 스크래치, 그래피티 같은 디테일을 더하는 데 사용됩니다.
- **Water Master Stack:** 반사, 굴절 및 기타 물 관련 시각 효과를 포함하여 수면을 시뮬레이션하는 데 특화된 셰이더입니다. 자세히 알아보려면 물 시스템 섹션을 참고하세요.

- **Fabric Master Stack:** 패브릭 머티리얼을 시뮬레이션합니다. 이 셰이더로 패브릭 머티리얼을 렌더링할 수 있으며, 광택 색상과 강도를 제어할 수 있습니다.



패브릭 머티리얼로 광택, 색상과 강도를 제어할 수 있습니다.

- **Eye Master Stack:** 각막, 공막 및 홍채를 포함하여 사실적인 눈을 렌더링할 수 있으며, 반사와 굴절을 세밀하게 제어할 수 있습니다.



Eye Master Stack은 사실적인 눈을 렌더링합니다.

- **Hair Master Stack:** 모발과 모피를 시뮬레이션하며 색상과 스페큘러 하이라이트를 제어하고 빛이 각 가닥을 통해 산란되는 방식을 정할 수 있습니다.



Hair Master Stack은 빛이 모발을 통과하며 산란되는 방식을 시뮬레이션합니다.

- **Canvas Master Stack:** 사용자 인터페이스 요소의 UGUI 셰이더로 사용하세요.
- **Fog Volume Master Stack:** 이 마스터 스택은 안개 볼륨을 위한 특정 설정과 컨텍스트를 제공합니다. 국지화된 연기나 연무와 같은 사실적인 환경 효과를 생성하는 데 사용하세요.
- **Fullscreen Master Stack:** 포스트 프로세싱 셰이더와 같은 전체 화면 효과에 사용하세요.



에너미즈 데모에는 디지털 휴먼 캐릭터가 등장합니다.

에너미즈 데모 프로젝트 또는 **Time Ghost: Character** 에셋에서 Eye Master Stack과 Hair Master Stack이 실제로 어떻게 사용되는지 살펴보세요.

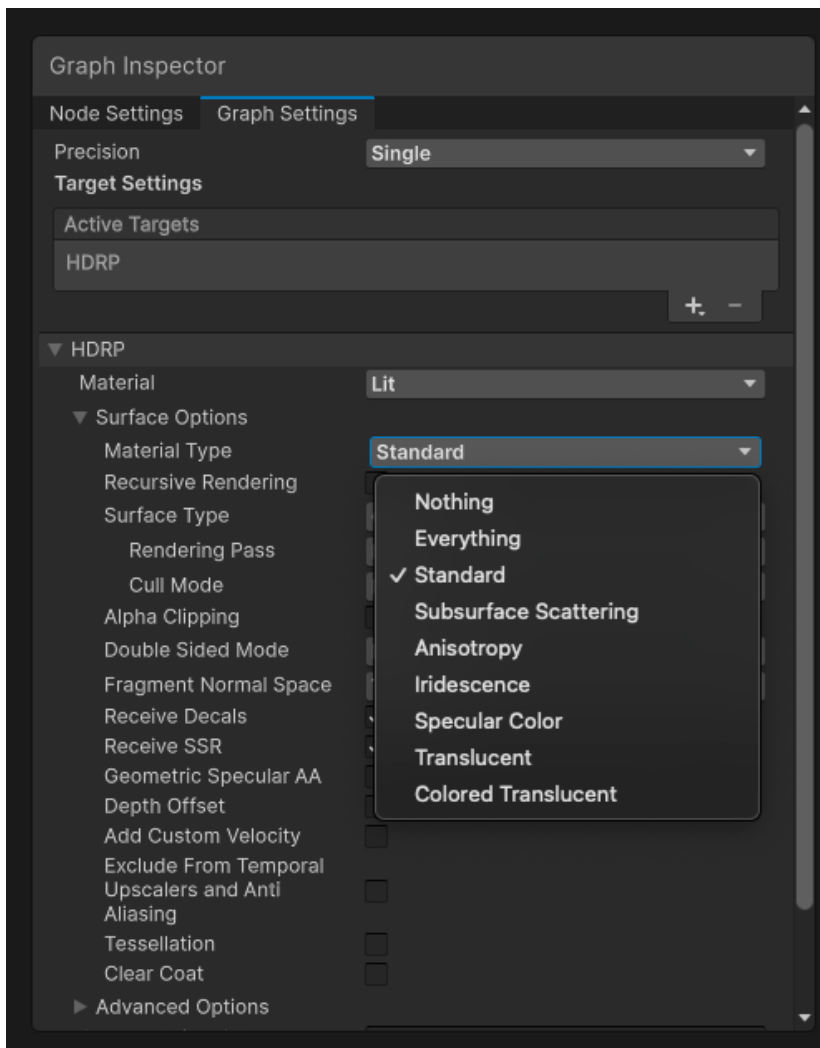
HDRP 전용 기능을 제공하는 셰이더 그래프 노드도 있습니다. Diffusion Profile, Emission, Exposure, HD Scene Color, HD Scene Depth 등의 노드입니다. 특히 Eye Master Stack이나 Water Master Stack에는 스택 전용 노드가 많이 있습니다. 기술 자료의 [HDRP 전용 노드](#) 전체 목록을 참고하세요.

Material Type 프로퍼티

Unity 6.1에서는 Shader Graph를 사용하여 셰이더 내에서 바로 **Material Type** 프로퍼티(Standard, Subsurface Scattering, Translucent 등)를 표시할 수 있습니다.

Shader Graph에서 어느 Material Type을 표시할지 지정하고 머티리얼의 인스펙터에서 Material Type을 동적으로 선택할 수 있습니다.

따라서 여러 Material Type과 머티리얼 배리언트에서 하나의 Shader Graph를 사용할 수 있습니다.



볼류메트릭 셰이더 그래프 안개

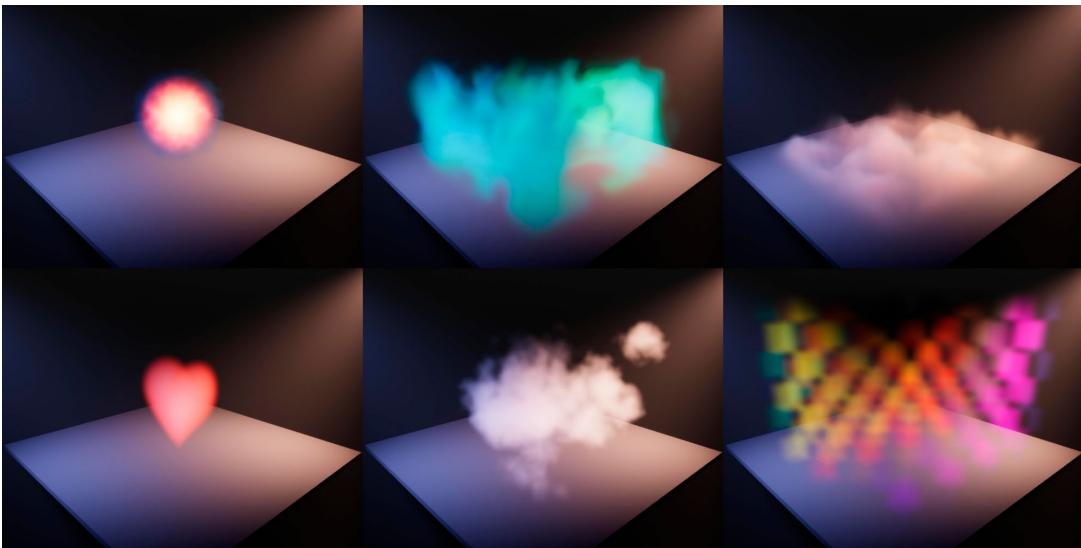
볼류메트릭 머티리얼로 고급 절차적 안개와 볼류메트릭 효과를 제작할 수 있습니다. 볼류메트릭 머티리얼은 Local Volumetric Fog 컴포넌트에 적용된 셰이더 그래프 조합을 사용합니다.



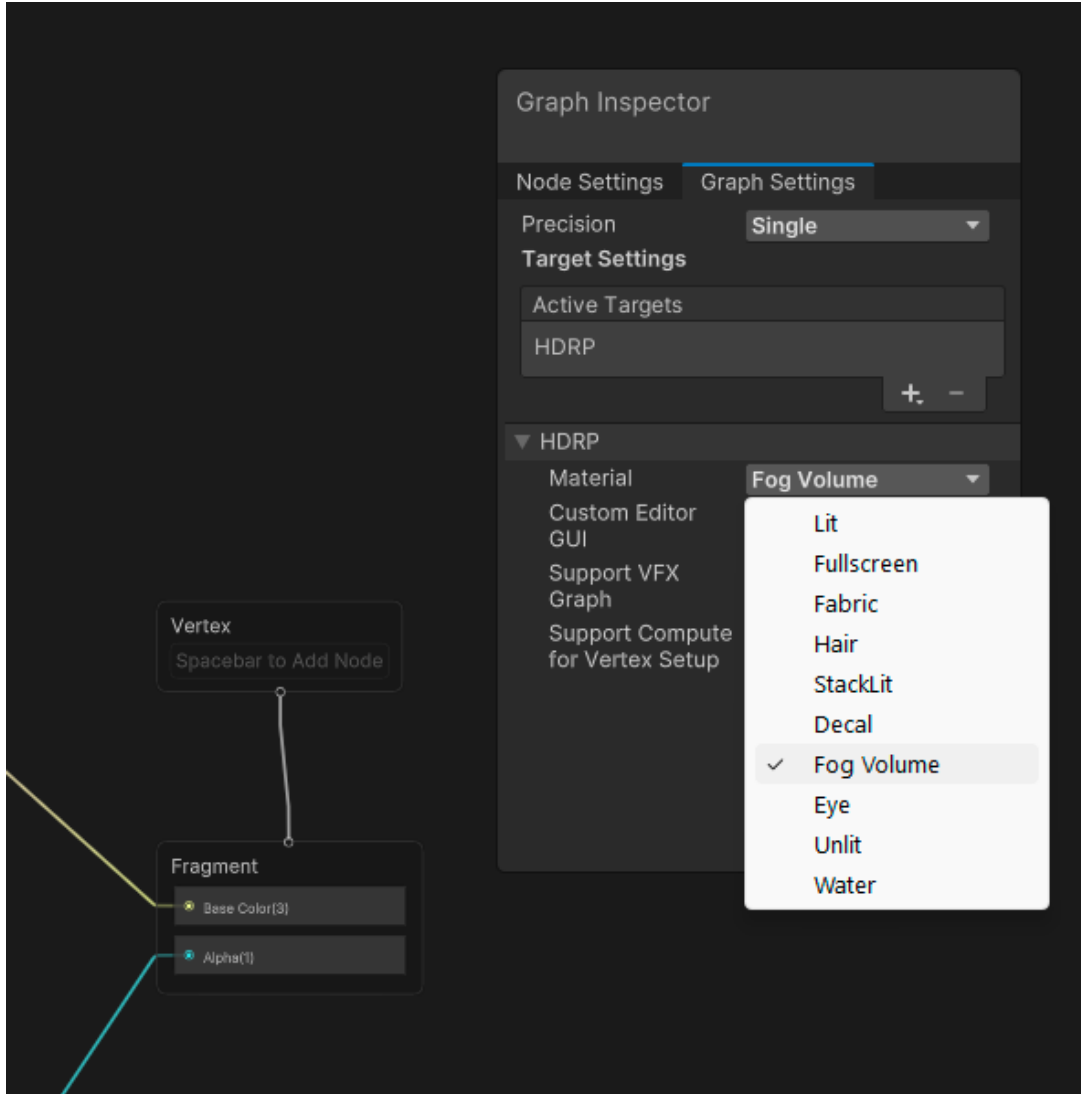
절차적 볼류메트릭 안개의 예시

이를 통해 커스텀 지면 안개나 구름 효과를 만들 수 있습니다. 또한 찌는 모래 폭풍이나 오로라 같은 다른 자연 현상도 구현할 수 있습니다.

더 많은 예시를 살펴보려면 패키지 관리자에서 **HDRP Volumetrics** 샘플을 임포트하세요. 이 샘플에서는 셰이더 그래프가 HDRP 머티리얼로 **Fog Volume**을 사용하여 Local Volumetric Fog 컴포넌트에 디테일과 애니메이션을 추가합니다.



HDRP 볼류메트릭 샘플 모음집은 패키지 관리자에서 다운로드할 수 있습니다.



Shader Graph 머티리얼로 Fog Volume을 선택합니다.

전체 화면 셰이더 그래프

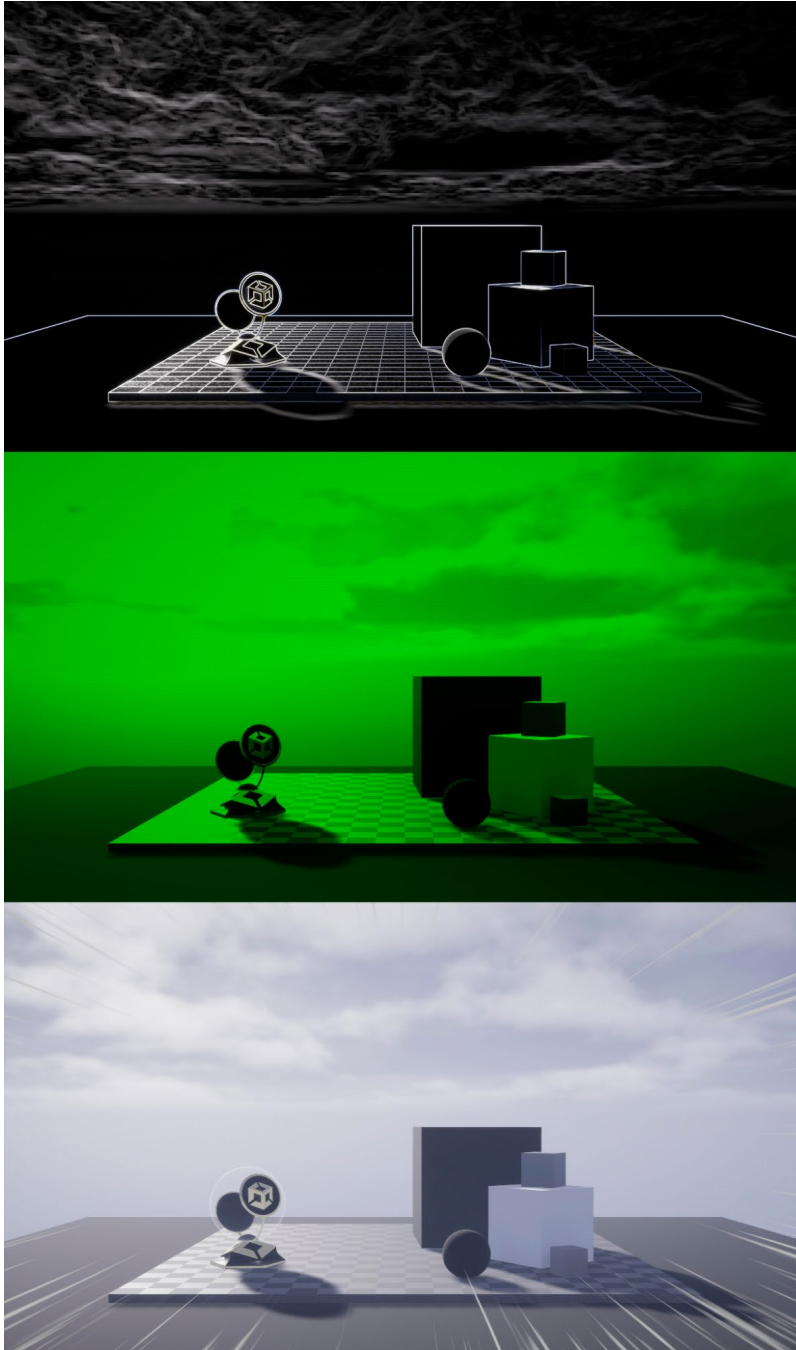
개발자와 아티스트는 Unity 전체 화면 셰이더를 활용하여 화면 부 전체에 적용되는 커스텀 효과를 제작할 수 있습니다. 이를테면 캐릭터가 대미지를 받을 때 화면이 빨간색으로 변하거나 화면에 물방울이 맺히는 등의 효과를 구현할 수 있습니다.

전체 화면 셰이더는 다음 세 가지 방법으로 사용할 수 있습니다.

- 커스텀 패스 효과 제작
- 커스텀 포스트 프로세싱 효과 제작
- C# 스크립트에서 **HDUtils.DrawFullscreen** 또는 **Graphics.Blit** 함수와 함께 사용

새 전체 화면 셰이더 그래프를 만들거나 기존 그래프를 수정하는 방식으로 전체 화면 셰이더를 생성할 수 있습니다. 이때 전체 화면 마스터 스택을 포함해야 합니다.

또한 HDRP에는 패키지 관리자를 통해 프로젝트에 임포트할 수 있는 전체 화면 셰이더 샘플이 포함되어 있습니다. 이 샘플들은 전체 화면 셰이더 그래프로 할 수 있는 작업을 보여 주는 예시입니다.

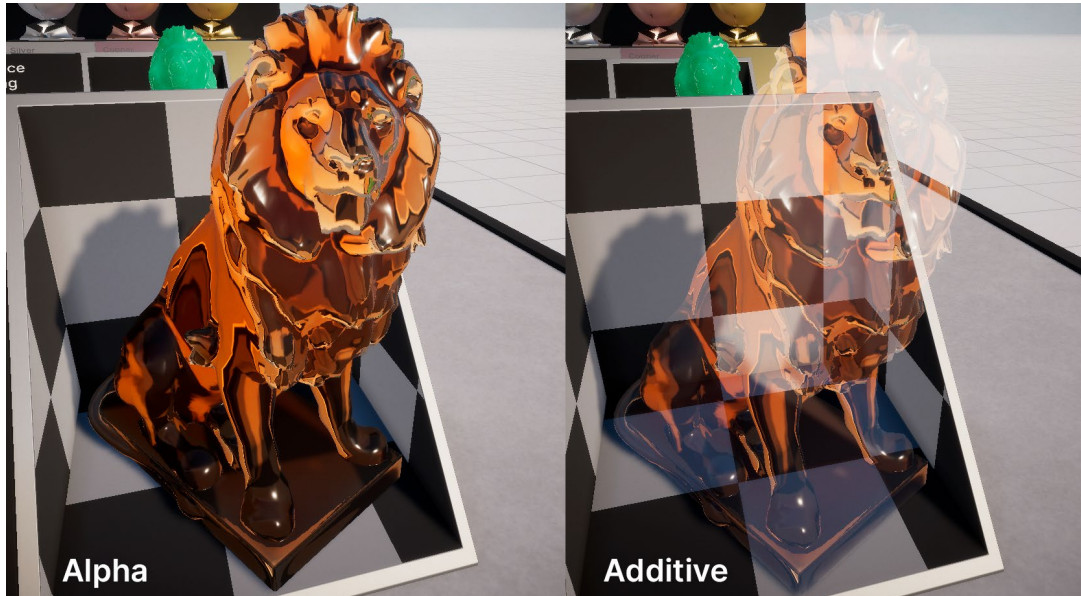


전체 화면 셰이더 그래프 예시

투명도

투명도는 유리나 투명 플라스틱과 같은 머티리얼에서 빛이 통과할 수 있는 정도를 나타냅니다. HDRP에서는 머티리얼 베이스 색상의 알파 값을 조정하여 투명도를 처리합니다. 알파 값이 낮을수록 머티리얼이 더 투명해지며, 알파 값이 1에 가까울수록 머티리얼이 불투명해집니다.

HDRP는 **Alpha**, **Premultiply**, **Additive**의 세 가지 투명도 모드를 지원합니다. Alpha 모드는 텍스처의 알파 값을 직접 사용하고, Premultiply 모드는 적용하기 전에 색상에 알파 값을 곱합니다. Additive 모드는 머티리얼의 색상을 배경에 추가합니다. 글로우 이펙트를 만들 때 유용할 수 있습니다.



Alpha 모드와 Additive 블렌딩 모드의 비교

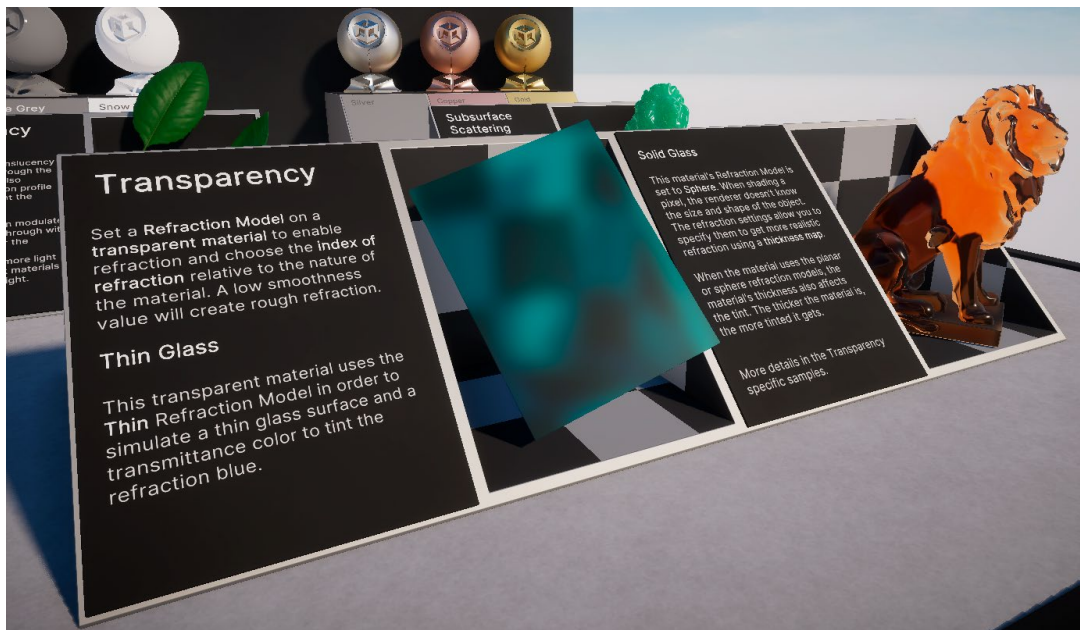
Lit Material 샘플

(패키지 관리자의) **Lit Material** 샘플에서는 투명 머티리얼을 설정하는 방법의 예시를 보여 줍니다. Transparent Surface Type을 사용할 때는 Refraction Model을 선택하고 머티리얼에 적합한 **Index of Refraction**을 설정합니다.

Thin Glass Refraction Model은 창문과 같은 얇은 유리 표면을 시뮬레이션하는 데 사용합니다. **Transmittance Color**는 굴절에 색조를 더합니다.

단단한 유리 오브젝트에는 **Sphere** 또는 **Planar** Refraction Model을 사용합니다. Sphere Refraction Model은 원석이나 유리구슬을 모방할 수 있습니다. Planar Refraction Model로는 얼음 큐브 같은 평평한 오브젝트를 만들 수 있습니다.

렌더러는 오브젝트의 모양과 크기를 알지 못하기 때문에, 이러한 모델에서 **Thickness Map**을 지정하면 더 사실적인 굴절을 구현할 수 있습니다. 맵의 머티리얼이 두꺼울수록 한층 더 색조를 띤 모습으로 보입니다.



Lit Material 샘플에서는 여러 Refraction Model을 확인할 수 있습니다.

Transparent Material 샘플

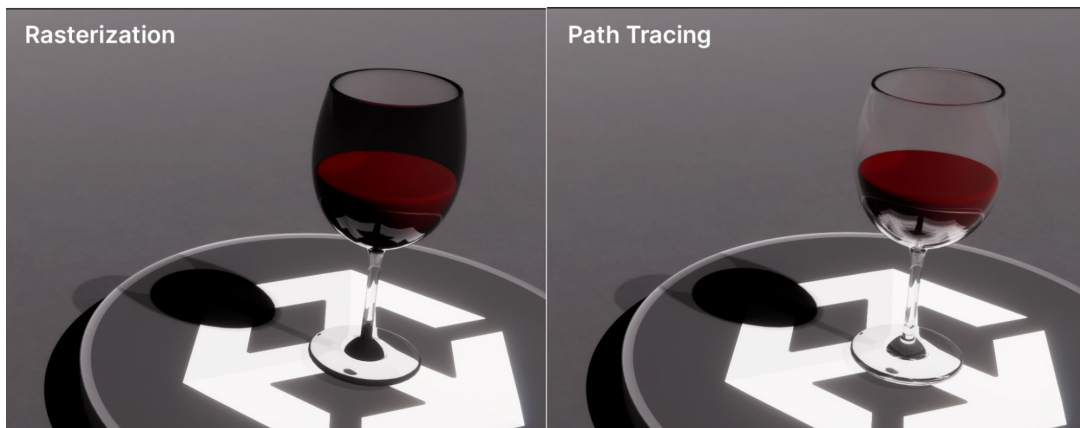
Unity 6에는 **Transparent** 샘플도 있습니다. 패키지 관리자에서 이용할 수 있는 이 샘플에서는 투명도 효과를 더 상세하게 살펴볼 수 있습니다. 이러한 머티리얼에서는 투명 표면을 사용할 때 그림자, 스택킹, 굴절을 설정하는 방법을 살펴볼 수 있습니다.



Transparent 샘플

씬에서 투명 머티리얼이 나타나는 방식에 영향을 주는 여러 렌더링 모드를 전환할 수 있습니다.

- **Rasterization:** 기본 래스터화 모드에서는 HDRP가 성능을 위해 물리적 정확도를 희생하고 굴절 효과를 모방합니다.
- **Recursive rendering:** 이 모드에는 투명 머티리얼 내부의 여러 빛 반사를 처리하여 더 사실적인 굴절과 내부 반사가 구현되어 있습니다. 기본 래스터화만으로는 충분하지 않은 복잡한 투명 오브젝트에 사용하세요.
- **Ray traced shadows:** 투명 머티리얼에는 표준 래스터화를 사용하지만, 그림자는 레이트레이싱으로 계산합니다. 이렇게 하면 전체 패스트레이싱 비용 없이도 투명 그림자가 더 정확하게 표현될 수 있습니다.
- **Path tracing:** 패스트레이싱은 빛이 투명 매질을 통과하는 방식을 시뮬레이션합니다. 가장 물리적으로 정확한 결과를 제공하지만 계산 비용이 매우 큼니다. 성능보다 품질이 중요한 경우에 사용하세요.



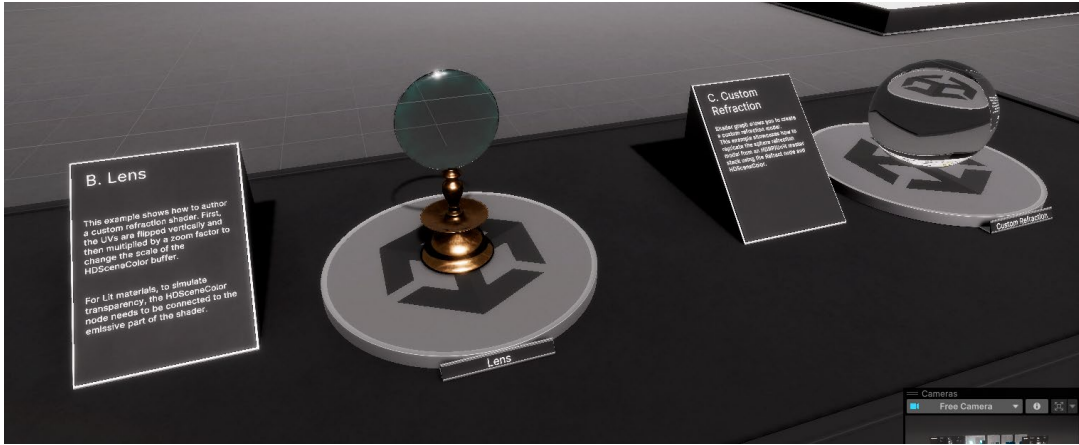
렌더링 모드 비교

Shader Graph 노드

기본적으로 Shader Graph에는 래스터화를 사용할 때(레이트레이싱은 호환되지 않음) 커스텀 투명도를 구현할 수 있는 몇 개의 노드가 있습니다. 이러한 노드는 굴절 및 뎁스 기반 투명도를 시뮬레이션하는 데 사용할 수 있습니다.

- **Refract:** 이 노드는 유리나 물과 같은 투명 머티리얼을 통과하면서 빛이 굴절되는 효과를 시뮬레이션합니다.
- **Scene Depth:** 이 노드는 투명 표면 뒤의 뎁스 정보를 캡처하여 뎁스 기반 투명도와 블렌딩에 사용합니다.
- **Scene Depth Difference:** 이 노드는 투명 오브젝트와 배경의 뎁스 차이를 계산합니다. 부드러운 경계나 왜곡 효과를 구현할 때 사용하세요.
- **HD Scene Color:** 이 노드는 배경 색상 데이터를 가져와서 다른 투명도, 블렌딩, 굴절 효과에 사용합니다.

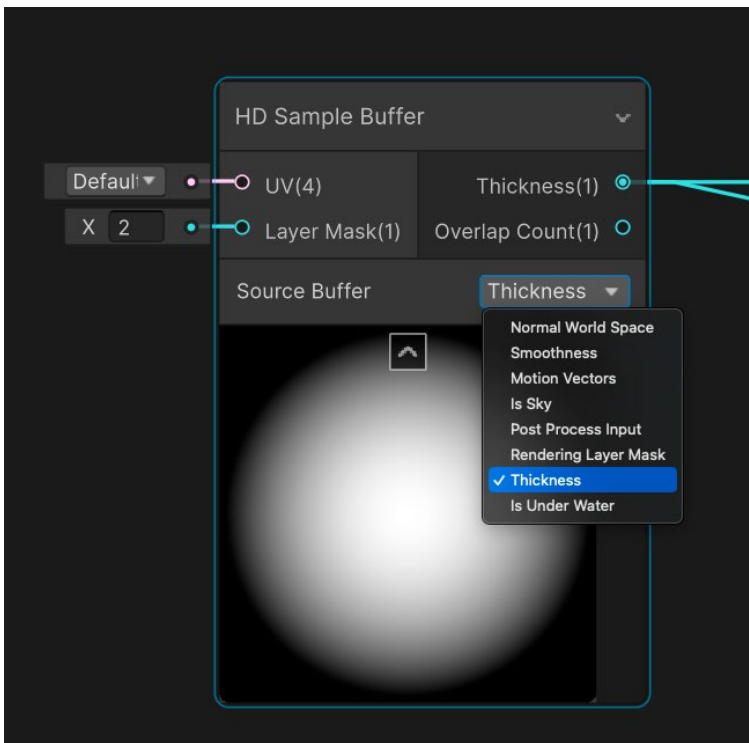
이 노드들을 조합하여 씬에서 커스텀 투명 효과를 만들 수 있습니다. 아래 예시는 UV와 HDSceneColor 버퍼를 조작하여 굴절 효과를 재현하는 방법을 보여 줍니다.



Lens 및 Custom Refraction 샘플

Compute Thickness

HDRP 에셋에서 **Compute Thickness**를 활성화하면 지정된 LayerMask 내 오브젝트의 두께 정보가 들어 있는 버퍼가 할당됩니다. Shader Graph에서 HD Sample Buffer 노드를 사용하여 이 버퍼를 샘플링하면 색상과 굴절 결과에 영향을 줄 수 있습니다.



HD Sample Buffer에서 Compute Thickness를 샘플링합니다.

예를 들어, 곰돌이 젤리에 적용된 x-ray 셰이더는 Compute Thickness 버퍼를 사용하여 곰돌이 젤리의 골격을 드러냅니다. Shader Graph는 인스펙터에서 이 효과의 외형과 모양을 바꿀 수 있는 추가 컨트롤을 제공합니다.



X-Ray 셰이더는 Compute Thickness 버퍼를 사용합니다.

피하 산란 및 반투명

SSS(피하 산란)는 반투명 머티리얼에서 발생하는 현상입니다. 반투명은 빛이 물질을 통과하되 그 과정에서 산란되는 특수한 유형의 투명도입니다.

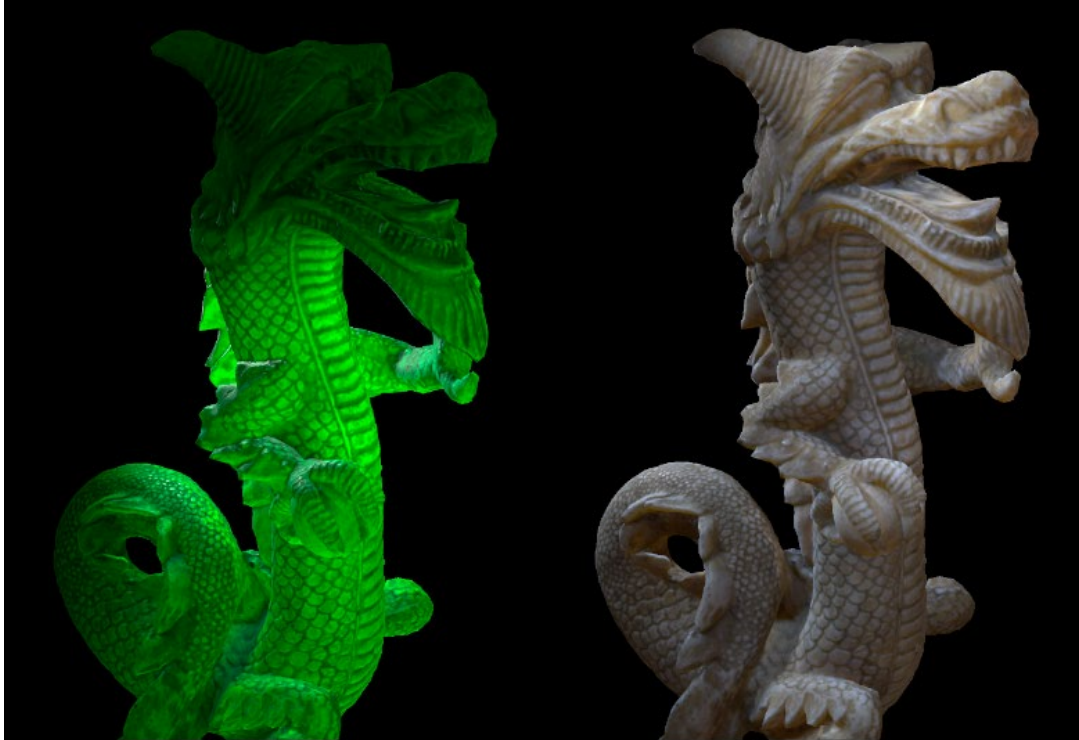
이 효과는 피부나 나뭇잎 같은 유기적인 머티리얼을 통과하는 빛에 부드러운 글로우를 적용하며, 머티리얼에 거친 플라스틱과 같은 질감 대신 부드러운 느낌을 선사합니다.

빛이 반투명 머티리얼에 닿으면 모두 반사되는 대신 일부는 안으로 통과하고 머티리얼 내부에서 반사되며 산란됩니다. 산란된 빛의 일부는 결국 들어온 곳과 다른 위치에서 머티리얼을 빠져나가게 됩니다.

HDRP는 스크린 공간 블러 기술을 사용하여 피하 산란을 구현합니다. 또한 게임 오브젝트의 뒤에서 빛이 통과하여 오브젝트가 투명해 보이게 만드는 빛의 투과도 처리합니다. 투과로 빛이 통과하는 과정에서 빛에 색상이 입혀질 수도 있습니다.

HDRP의 SSS에는 두 가지 머티리얼 유형이 있습니다.

- **Subsurface Scattering**은 스크린 공간 블러 효과와 투과를 모두 구현합니다.
- **Translucent**는 투과만 구현합니다.



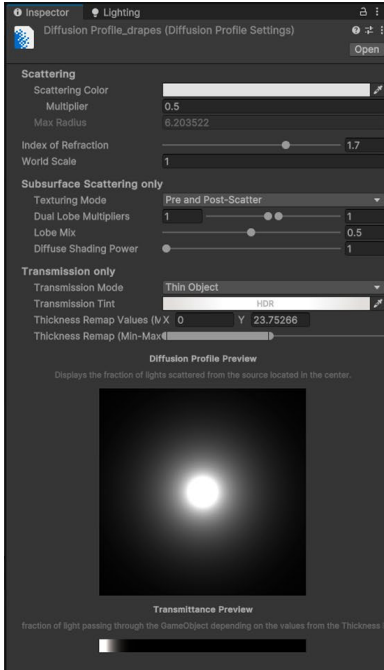
Translucency

**Subsurface
scattering**

Translucency는 투과만 구현합니다. Subsurface Scattering은 스크린 공간 블러 효과를 더합니다.

HDRP 에셋과 **Project Settings > HDRP Default Settings**에서 피하 산란을 활성화합니다. 그다음 SSS 설정을 저장하는 [Diffusion Profile](#) 에셋을 생성합니다. HDRP는 뷰에서 동시에 최대 15개의 커스텀 프로파일을 지원하며, 오버라이드 옵션도 제공합니다.

Material Type을 필요에 따라 **Subsurface Scattering** 또는 **Translucent**로 설정하고 Diffusion Profile을 할당합니다.



이 씬의 커튼 같은 SSS 머티리얼은 확산 프로파일을 사용합니다.

Diffusion Profile은 산란된 빛의 색상, 투과 틸트, 두께 등의 프로퍼티를 제어합니다.

각 머티리얼에는 SSS에 영향을 주는 다음 두 가지 옵션이 있습니다.

- **Thickness Map**은 오브젝트의 여러 부분에 적용되는 산란과 투과 효과를 제어합니다.
- **Transmission Mask**는 투과의 전반적인 강도를 제어합니다. 예를 들어 나무의 몸통이나 앞에 하나의 셰이더를 사용할 수 있습니다.



피하 산란을 사용하면 나뭇잎을 더 사실적으로 렌더링할 수 있습니다.

SpeedTree 및 Transmission Mask

Transmission Mask는 SpeedTree 초목에서 이파리에는 반투명 효과를 적용하고 나무 껍질과 잔가지는 불투명하게 유지되도록 하는 데 유용합니다. Transmission Mask를 사용하면 피하 산란이 존재해서는 안 되는 영역에 적용되는 것을 방지할 수 있습니다.

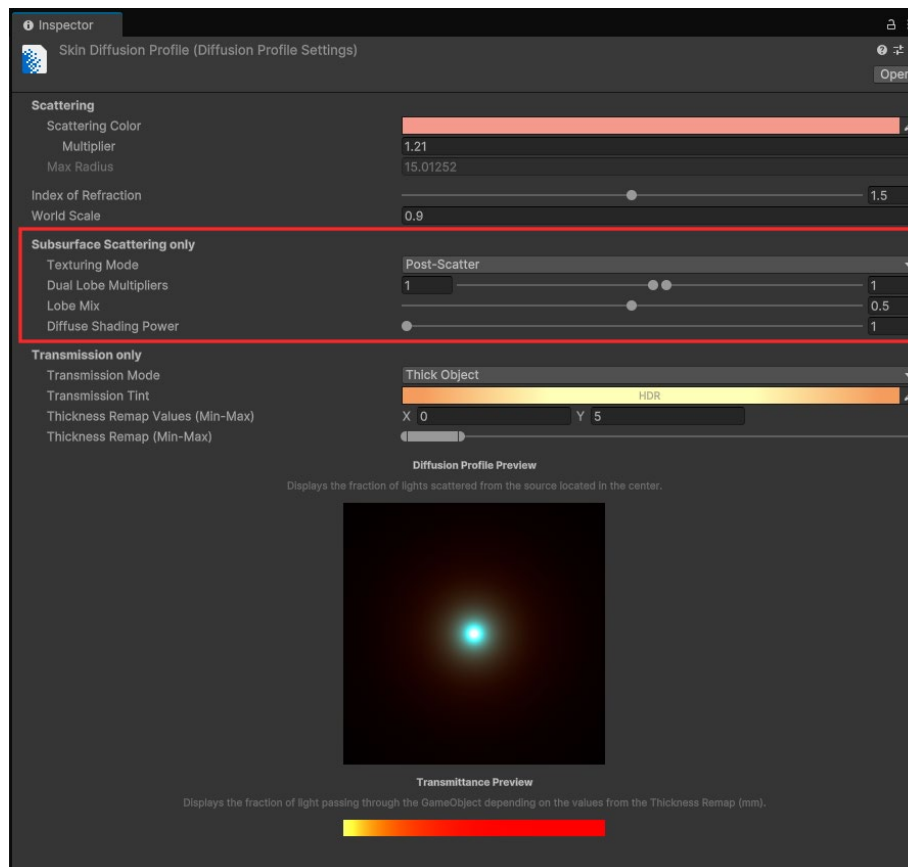
[SpeedTree 8 하위 그래프 에셋](#)(HDRP/Nature/SpeedTree8.shadergraph)은 Transmission Mask 노드에 Subsurface Map을 사용하여 나무의 껍질과 잔가지에서 의도하지 않은 빛 투과를 제거합니다. 또한 이를 통해 3D 지오메트리의 조명과 어울리지 않게 너무 밝았던 빌보드 조명 문제를 해결합니다.

Unity 6의 새로운 기능: 향상된 피부 렌더링

반투명 머티리얼의 [Diffusion Profile](#)은 표면 내에서 빛이 산란되는 방식을 제어합니다. Unity 6에서는 피부 렌더링을 개선하기 위해 Diffusion Profile에 새로운 파라미터가 추가되었습니다.

- **Dual Lobe** 스펙큘러 반사는 피부 표면의 얇은 유분층을 시뮬레이션하여 유분의 선명한 하이라이트와 더 깊은 층의 부드러운 반사를 구현합니다.
- **Diffuse Power**는 피부처럼 강한 피하 산란을 갖는 머티리얼에서 빛이 퍼지는 방식을 개선합니다.

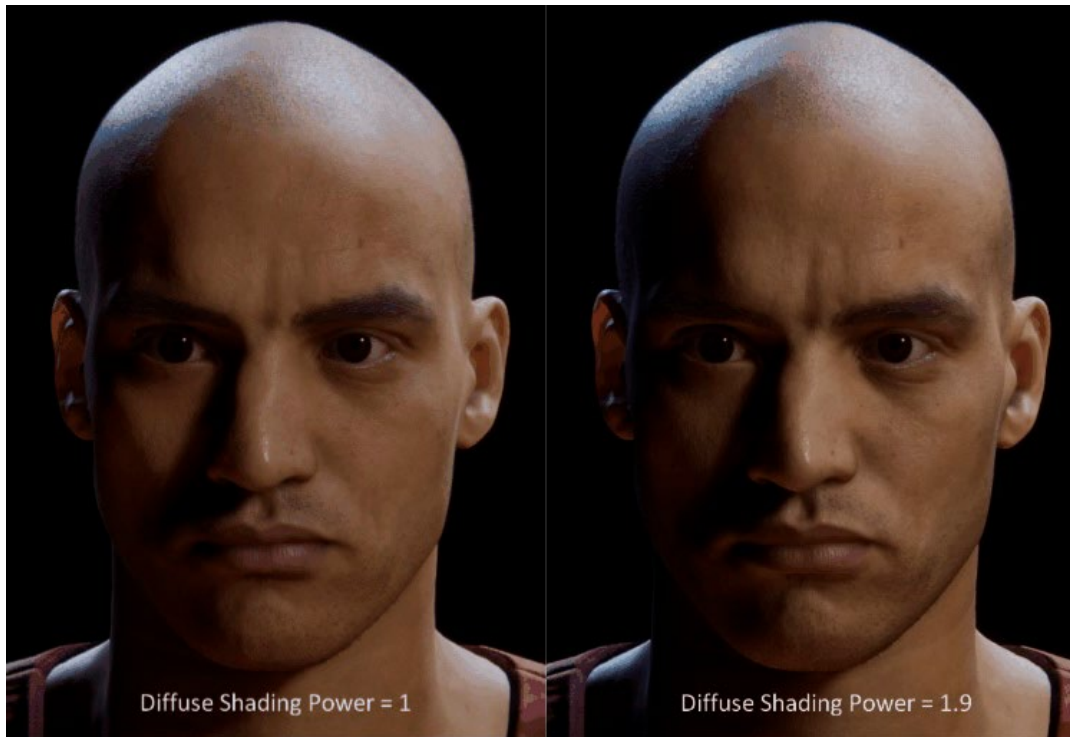
이러한 업데이트는 다양한 조명 조건에서 피부 머티리얼을 더 자연스럽게 보이도록 합니다.



Unity 6에서는 Diffusion Profile에 파라미터가 추가되었습니다.



Dual Lobe는 사람 피부의 스페큘러 반사를 구현합니다.



Diffuse Shading Power는 피부 머티리얼에서 빛이 퍼지는 방식을 제어합니다.



피하 산란은 피부를 더 사실적으로 렌더링합니다.

데칼

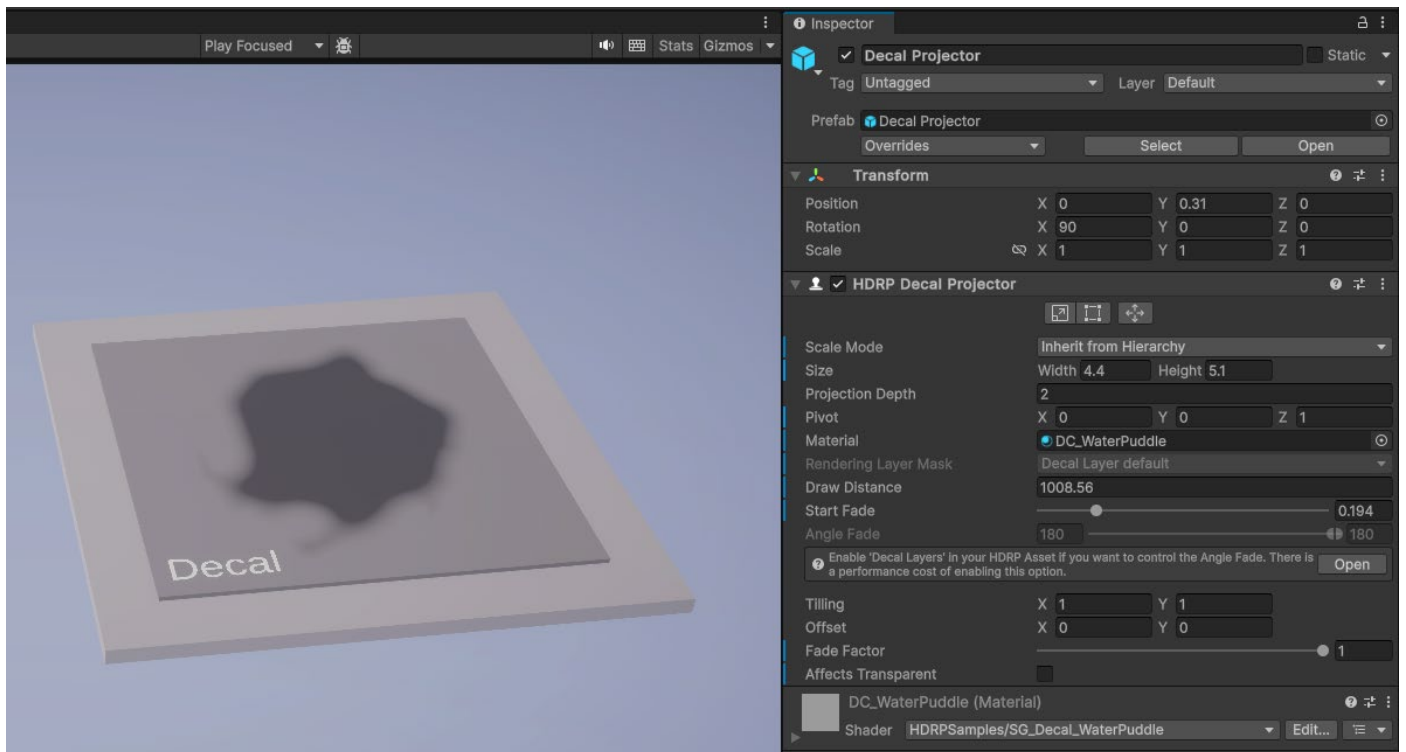
데칼은 **데칼 셰이더** 또는 데칼 마스터 스택을 사용하는 머티리얼입니다. 데칼은 지오메트리를 추가하지 않으면서 머티리얼과 표면에 그래픽 오버레이로 디테일을 더합니다.

기본적으로 데칼은 씬의 다른 오브젝트에 투영되는 텍스처로, 평평한 이미지(투영된 데칼) 또는 3D 오브젝트의 셰이프를 따르는 형태(메시 데칼)로 구현됩니다.

Decal Projector 컴포넌트는 데칼 제작에 사용되는 기본 툴입니다. 이 컴포넌트로 데칼의 크기, 뎀스, 기타 프로퍼티를 제어할 수 있습니다.

Start Fade와 **Draw Distance**로 거리에 따른 데칼의 가시성을 제어할 수 있습니다. **Angle Fade**를 활성화하면 벡터스 노멀에 대한 데칼의 방향을 기준으로 데칼의 페이드아웃을 설정할 수 있습니다.

데칼을 사용하면 도시 벽 위에 그래피티를 그리거나, 바닥과 기계의 마모된 부분을 표현하거나, 차량과 사물에 패턴이나 엠블럼을 추가하는 등 다양한 작업을 할 수 있습니다.



데칼로 디테일을 추가할 수 있습니다.

Unity 6.1에서는 Shader Graph로 만든 데칼이 투명 오브젝트에도 영향을 줍니다. 이를 활용하면 빗방울, 잔물결, 커스텀 각인, 유리에 묻은 먼지 효과 등과 같은 절차적 효과를 만들 수 있습니다.

셰이더: 모피와 모발

Hair 셰이더 또는 Hair Shader Graph는 HDRP에서 모발과 모피를 렌더링하기 위한 시작점으로 사용할 수 있습니다. Hair 셰이더와 Hair Shader Graph는 사실적인 모발 효과를 만들기 위해 '헤어 카드'라는 레이어를 사용합니다.

각 헤어 카드는 머리카락의 서로 다른 영역을 나타냅니다. 반투명 헤어 카드를 사용하는 경우, 어느 시점에서 보든 뒤에서 앞 순서로 보이도록 수동으로 정렬해야 합니다.

Hair Master Stack은 모발과 모피에 사용할 수 있는 두 가지 머티리얼 유형을 지원합니다.

- **Approximate Material Type:** 이 모발 머티리얼 유형은 헤어 카드 지오메트리에 적합하며 Kajiya-Kay 헤어 모델을 기반으로 합니다. 이 유형을 사용할 때는 씬의 조명 환경에 맞게 Fragment 컨텍스트의 블록을 조정해야 합니다.
- **Physical Material Type:** 이 모발 머티리얼 유형은 가는 원통이나 끈 형태의 머리카락 지오메트리에 적합하며 Marschner 헤어 모델을 기반으로 합니다. 이 유형은 모든 조명 환경에 적합합니다.

머리카락 기반 모발은 더 정교하고 사실적이지만 많은 계산 리소스가 요구됩니다. 따라서 메인 캐릭터나 고품질 클로즈업에 적합합니다. Physical Material Type은 [다중 산란](#)도 지원하여 머리카락 내에서의 사실적인 빛 상호 작용을 시뮬레이션합니다.

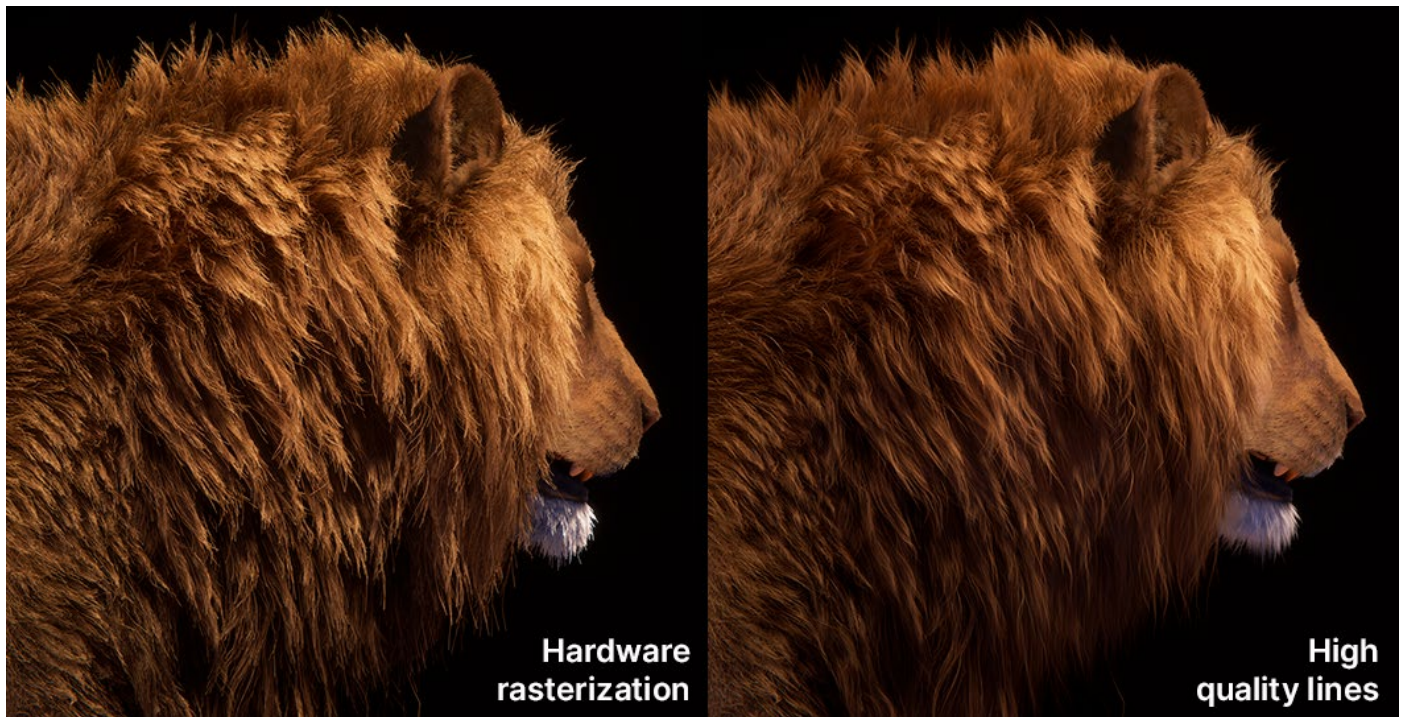


HDRP에서는 사실적인 모발과 모피를 생성할 수 있습니다.

Unity 6.1의 새로운 기능:

- **High Quality Line Rendering:** 기존 하드웨어 래스터화로 얇은 선을 렌더링하면 특히 모발과 모피에서 앨리어싱과 정렬 문제가 발생할 수 있습니다. High Quality Line Rendering은 더 나은 투명도와 안티앨리어싱을 갖춘 고품질 선 렌더링을 제공합니다. 자세한 내용은 [고품질 선 렌더링 사용](#)을 참조하세요.
- **Screen Coverage LOD Mode:** 이 모드는 스크린 공간에서 차지하는 면적에 따라 렌더링할 머리칼 수를 동적으로 조정하여 성능을 향상합니다.
- **성능 개선:** 머리칼 기반 모발 렌더링은 많은 머리칼을 렌더링할 때 성능, 비주얼 품질, 안티앨리어싱을 최적화할 수 있는 소프트웨어 래스터라이저를 사용합니다.

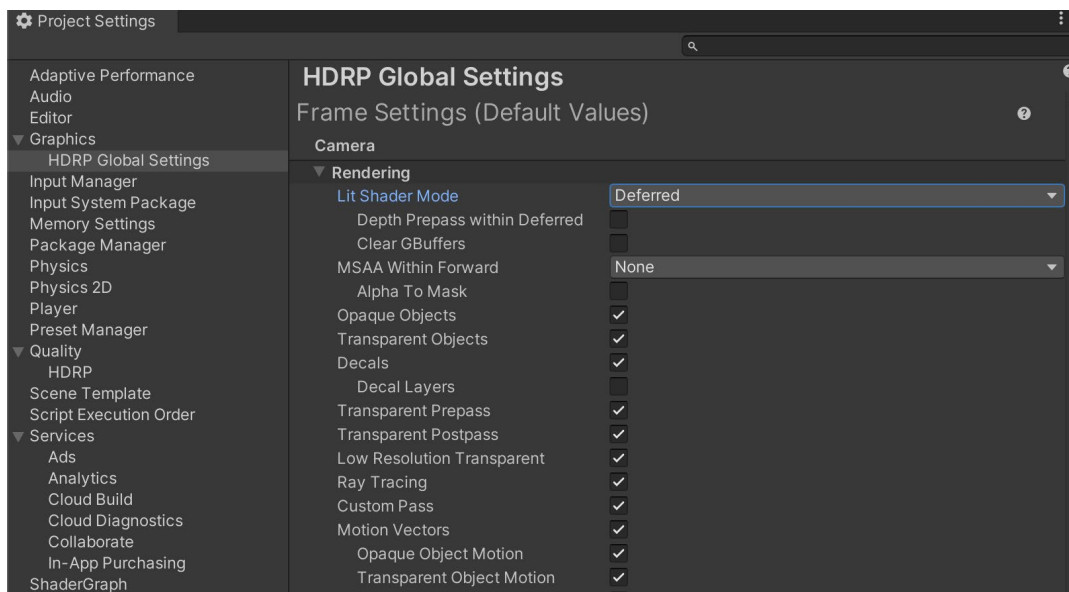
자세한 내용은 [모발 및 모피](#) 기술 자료 페이지를 참조하세요. Unity에서 모발이나 모피 설정이 처음이라면 [모발 시뮬레이션 시작하기](#)로 모발 시스템을 시작해 보세요.



High Quality Line Rendering은 더 나은 투명도와 안티앨리어싱으로 렌더링합니다.

포워드 렌더링 및 디퍼드 렌더링 비교

파이프라인 에셋에서 HDRP를 설정할 때 보통 **Rendering**에 있는 **Lit Shader Mode**부터 설정하게 됩니다. **Deferred, Forward, Both** 중에 선택할 수 있습니다. 각 옵션은 파이프라인에서 지오메트리에 조명을 적용하고 렌더링하는 방식과 관련된 일련의 특정 작업인 렌더링 경로를 나타냅니다.



기본 HDRP 설정 수정

렌더링 경로 커스터마이징

Lit Shader Mode에서 **Forward** 또는 **Deferred**를 선택하여 기본 렌더링 경로를 설정합니다.

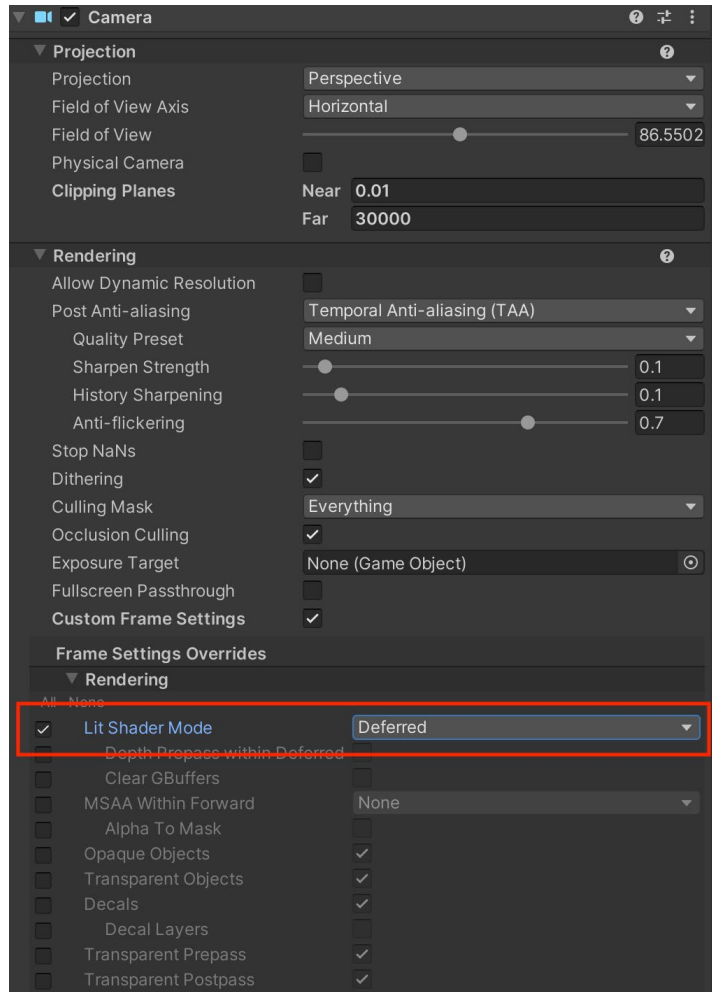
HDRP는 유연성이 높아서 **Both**를 선택할 수도 있습니다. 이 옵션을 사용하면 대부분의 렌더링에 한 렌더링 경로를 사용한 다음 카메라별로 렌더링 경로를 오버라이드합니다. 그러나 이 방식을 사용하면 GPU 메모리가 더 많이 소모됩니다. 대부분의 경우 Forward 또는 Deferred 중 하나를 선택하는 것이 좋습니다.

- 모든 카메라의 기본 설정을 변경하려면 **HDRP Default Settings**로 이동하여 **Default Frame Settings**를 찾습니다. 여기에서는 **Camera, Baked or Custom Reflection, Realtime Reflections** 등을 설정할 수 있습니다.

Rendering 그룹의 **Lit Shader Mode**에서 렌더링 경로를 설정합니다.

- 특정 카메라에 대한 설정을 오버라이드하려면 **Custom Frame Settings**를 선택합니다.

그런 다음 **Rendering** 그룹에서 **Lit Shader Mode**의 렌더링 경로를 오버라이드해 변경합니다.



카메라의 커스텀 프레임 설정 수정

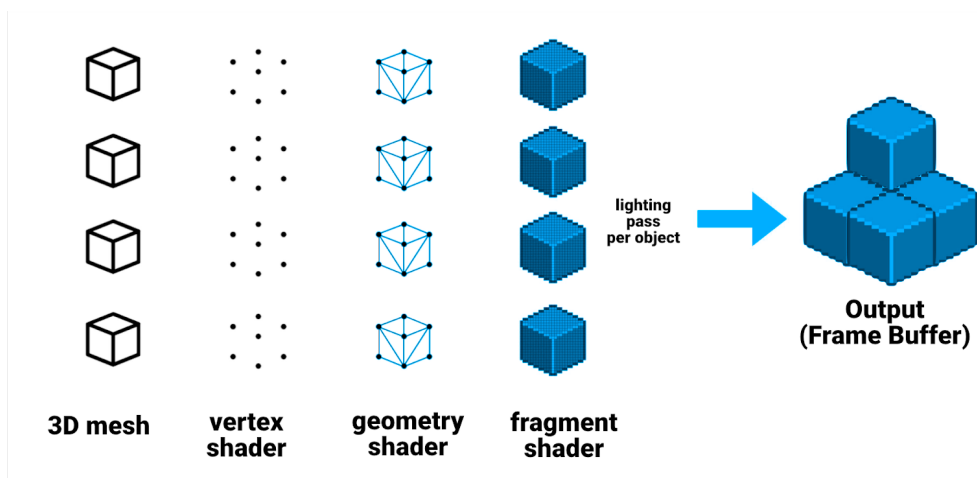
📌 렌더링 경로에 대한 자세한 내용

렌더링 경로가 어떻게 작동하는지 이해하면 Lit Shader Mode가 파이프라인에서 다른 설정에 어떤 영향을 미치는지 파악할 수 있습니다.

포워드 렌더링

포워드 렌더링에서 그래픽 카드는 화면의 지오메트리를 버텍스로 분할합니다. 버텍스는 더 나아가 프래그먼트 또는 픽셀로 세분화를 거쳐 화면에 렌더링되며 최종 이미지를 생성합니다.

각 오브젝트는 한 번에 하나씩 그래픽스 API로 전달됩니다. 포워드 렌더링은 각 광원마다 리소스를 소모합니다. 씬에 광원이 많을수록 렌더링 시간도 길어집니다.



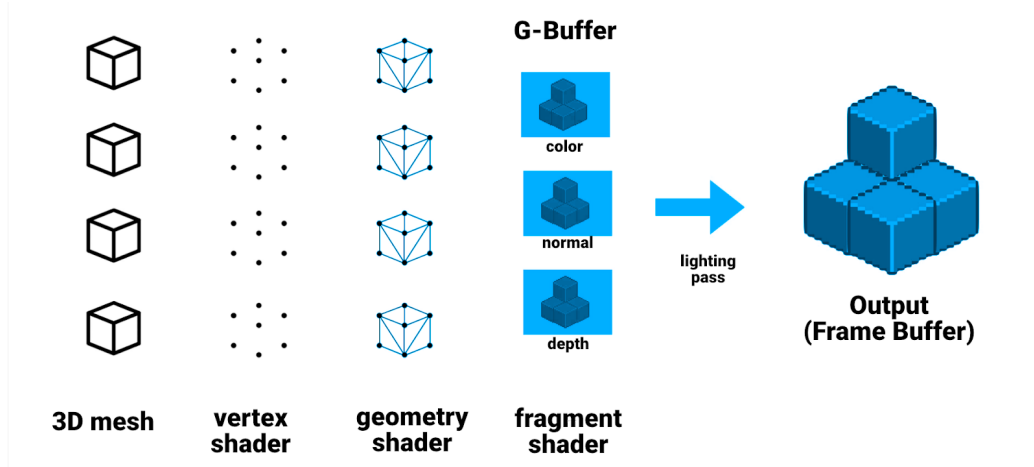
포워드 렌더링 경로

포워드 렌더링은 별도의 패스에서 광원을 드로우합니다. 같은 게임 오브젝트에 여러 광원이 닿는 경우 엄청난 오버드로우를 유발할 수 있으므로, 광원과 오브젝트가 많으면 속도가 느려질 수 있습니다.

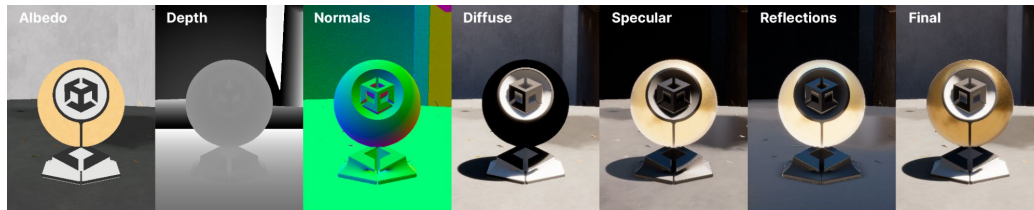
기존 포워드 렌더링과 달리 HDRP는 포워드 렌더러에 몇 가지 효율성을 더했습니다. 예를 들어 오브젝트 머티리얼당 하나의 패스에서 여러 광원을 함께 컬링하고 렌더링합니다. 그럼에도 불구하고 상대적으로 리소스를 많이 사용하는 프로세스입니다. 성능이 중요한 경우 디퍼드 셰이딩을 대신 사용하는 것이 좋습니다.

디퍼드 셰이딩

HDRP는 조명이 오브젝트별로 계산되지 않는 디퍼드 셰이딩을 사용할 수도 있습니다. 디퍼드 셰이딩은 무거운 렌더링을 후순위로 연기하고 두 개의 패스를 사용합니다.



디퍼드 셰이딩 경로



디퍼드 셰이딩은 각 오브젝트 대신 버퍼에 조명을 적용합니다. 각 패스는 최종 렌더링된 이미지에 영향을 줍니다.

Unity는 첫 번째 패스, 즉 **G버퍼** 지오메트리 패스에서 게임 오브젝트를 렌더링합니다. 이 패스는 여러 유형의 지오메트리 프로퍼티를 가져오고 이를 디퓨즈 색상, 스페큘러 컬러, 표면 평활도, 오클루전, 노멀 등의 텍스처 세트에 저장합니다.

이후 두 번째 패스, 즉 **조명 패스**에서 G버퍼가 완료된 후 씬의 조명을 렌더링합니다. 따라서 디퍼드 셰이딩은 셰이딩을 연기하는 셈입니다. 디퍼드 셰이딩 경로는 개별 오브젝트 대신 버퍼를 기반으로 각 픽셀 작업을 반복하고 조명 정보를 계산합니다.

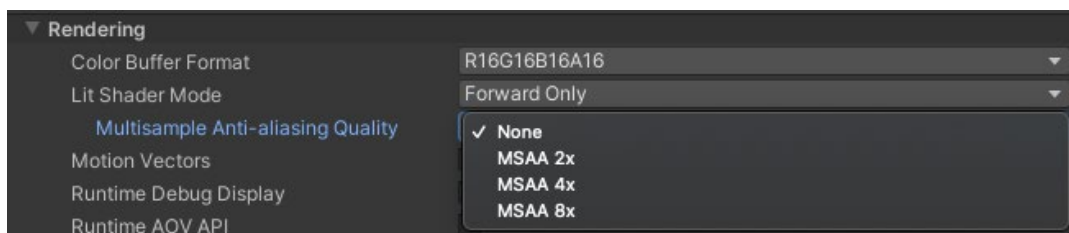
렌더링 경로 간의 기술적인 차이점에 대한 자세한 내용은 HDRP 기술 자료의 [포워드 렌더링 및 디퍼드 렌더링](#)을 참조하시기 바랍니다.

안티앨리어싱

Lit Shader Mode의 렌더링 경로는 작업자가 안티앨리어싱을 사용하여 렌더링에서 들쭉날쭉한 모서리를 없애는 방식에 영향을 줍니다. HDRP에서는 제작 중 필요에 따라 몇 가지 안티앨리어싱 기술을 사용할 수 있습니다.

MSAA(멀티 샘플 안티앨리어싱)

MSAA(멀티 샘플 안티앨리어싱)는 PC 게이머 사이에서 널리 사용되는 안티앨리어싱 방법입니다. MSAA는 개별 폴리곤의 모서리를 다듬는 고품질 하드웨어 기술이며 Unity에서는 포워드 렌더링에서만 작동합니다. 대부분의 최신 **GPU**는 2x, 4x, 8x MSAA 샘플을 지원합니다.



MSAA 품질 설정

활성화된 파이프라인 에셋에서 Lit Shader Mode를 **Forward Only**로 설정합니다. 이어서 **Multisample Anti-aliasing Quality**에서 **MSAA 2x**, **MSAA 4x**, **MSAA 8x** 중 하나를 선택합니다. 값이 높을수록 안티앨리어싱은 개선되지만 속도는 느려집니다.



카메라 뷰를 확대하면 차이를 더 명확하게 확인할 수 있습니다.



원본 씬



MSAA 설정이 적용된 이미지

다음과 같은 제약에는 주의하세요.

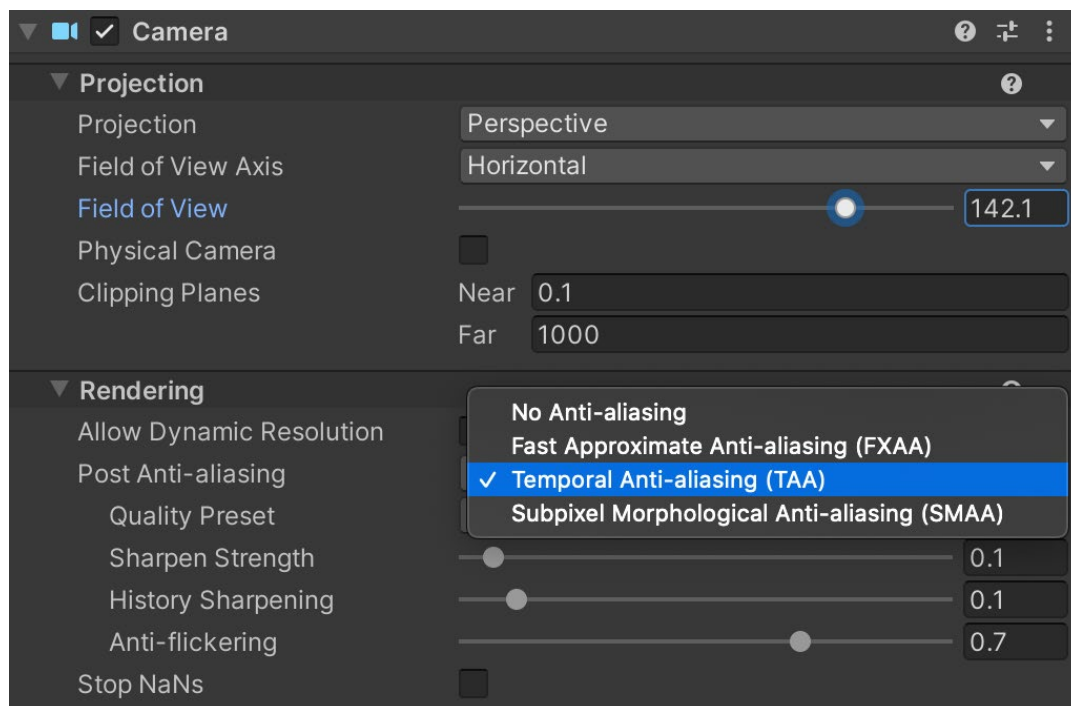
- MSAA는 씬 지오메트리를 텍스처 형태로 저장하는 디퍼드 셰이딩의 G버퍼와 호환되지 않습니다. 따라서 디퍼드 셰이딩을 사용하려면 포스트 프로세싱 안티앨리어싱 기술(아래 참조) 중 하나를 사용해야 합니다.

- MSAA는 폴리곤 에지 앨리어싱만 처리하기 때문에 선명한 스페큘러 조명을 받는 특정 텍스처와 머티리얼에서 발생하는 앨리어싱을 방지할 수 없습니다. 이러한 점이 문제가 된다면 MSAA를 다른 포스트 프로세싱 안티앨리어싱 기법(아래)과 함께 사용해야 할 수 있습니다.

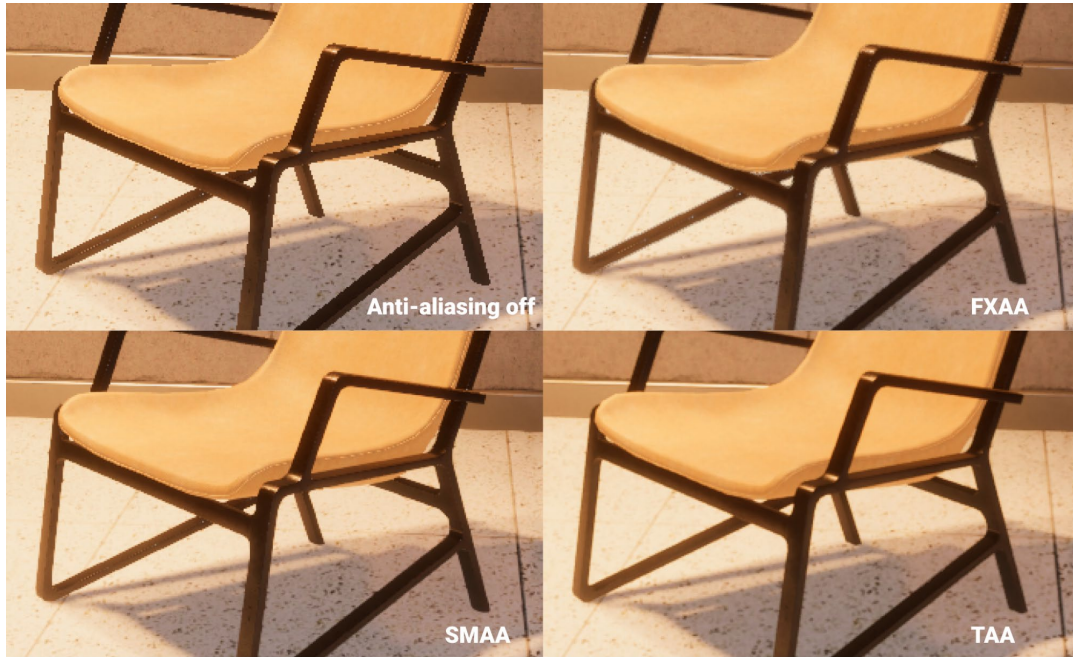
포스트 프로세싱 안티앨리어싱

카메라의 포스트 안티앨리어싱 설정을 통해서도 포스트 프로세싱 기법을 활용해 안티앨리어싱을 적용할 수 있습니다.

- **Temporal anti-aliasing (TAA)**은 지난 프레임과 현재 프레임의 정보를 결합하여 현재 프레임에서 **계단 현상**을 제거합니다. TAA를 사용하려면 **모션 벡터**를 활성화해야 합니다. TAA를 사용하면 일반적으로 훌륭한 결과를 얻을 수 있지만, 대비를 이루는 표면 앞에서 게임 오브젝트가 빠르게 이동하는 등의 일부 상황에서는 고스팅 결함이 나타날 수 있습니다. HDRP10에는 전형적인 TAA 결함을 줄이기 위한 개선이 이뤄졌습니다. 이를 통해 고스팅 결함을 줄이고 선명도를 개선할 수 있으며, 다른 솔루션에서 나타날 수 있는 깜빡이는 현상을 막을 수 있습니다.
- **Fast approximate anti-aliasing (FXAA)**은 고대비 영역 간에 픽셀을 블렌딩하는 **스크린 공간 안티앨리어싱** 알고리즘입니다. 엄청난 컴퓨팅 성능을 요구하지 않으며 상대적으로 빠른 기술이지만, 이미지의 전반적인 선명도를 낮출 수 있습니다.
- **Subpixel morphological anti-aliasing (SMAA)**은 이미지 테두리를 감지한 다음 블렌딩할 특정 패턴을 찾습니다. 그러면 FXAA보다 더 선명한 결과를 얻을 수 있으며, 이는 평평한 카툰풍 아트 스타일이나 깔끔한 아트 스타일에 적합합니다.



디퍼드 셰이딩을 사용하는 경우 카메라에서 포스트 안티앨리어싱을 조정합니다.



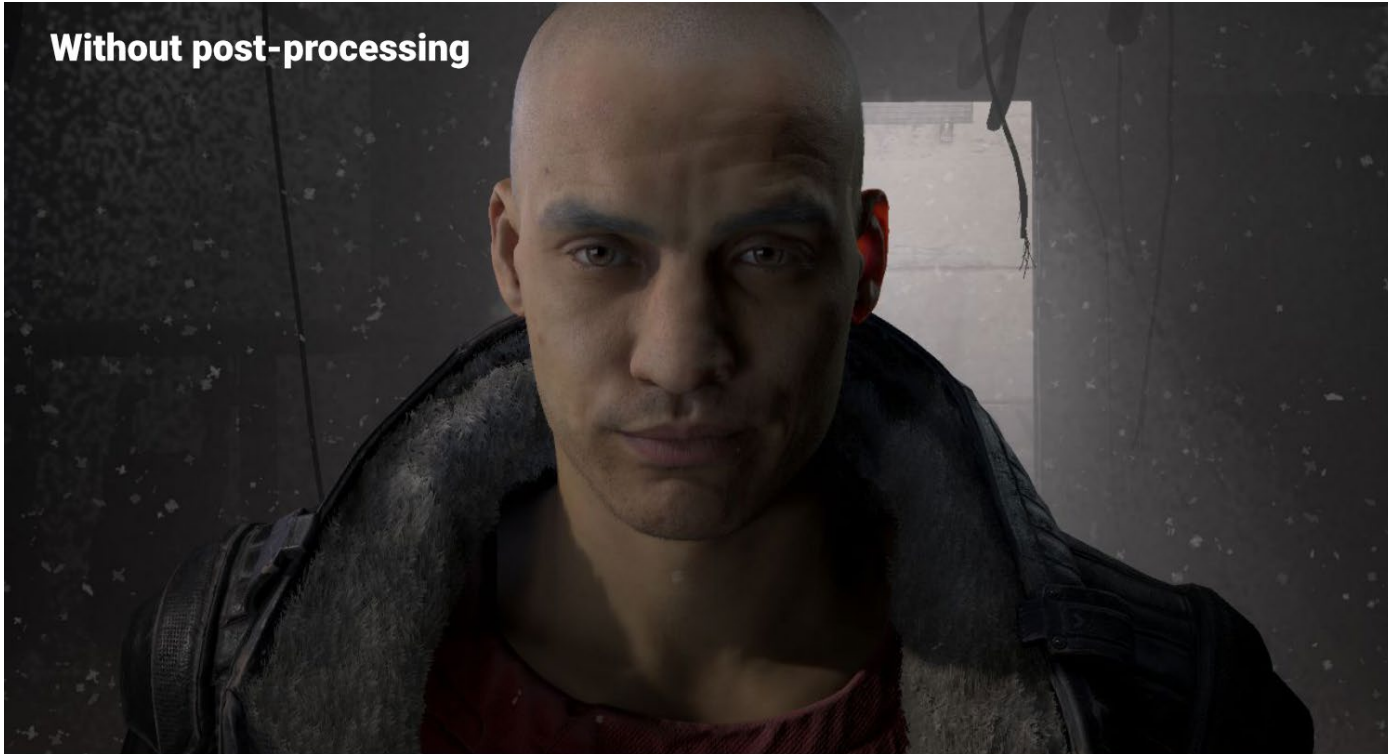
FXAA, SMAA, TAA 등의 포스트 프로세싱 안티앨리어싱 설정을 비교해 보세요.

참고: 포스트 프로세싱 안티앨리어싱을 MSAA와 함께 사용할 경우 렌더링 비용에 유의해야 합니다. 언제나 그렇듯이, 화질과 성능이 균형을 이루도록 프로젝트를 최적화하세요.

포스트 프로세싱

포스트 프로세싱 없이는 최신 하이엔드 그래픽스를 완성할 수 없습니다. 모든 것을 포스트 프로세싱에 기댈 수는 없겠으나, 이미지를 더 영화처럼 만드는 필터와 전체 화면 이미지 효과가 적용되지 않은 렌더링 이미지는 상상하기 어렵습니다. 따라서 HDRP는 자체적으로 빌트인된 포스트 프로세싱 효과를 번들로 제공합니다.

Without post-processing



With post-processing

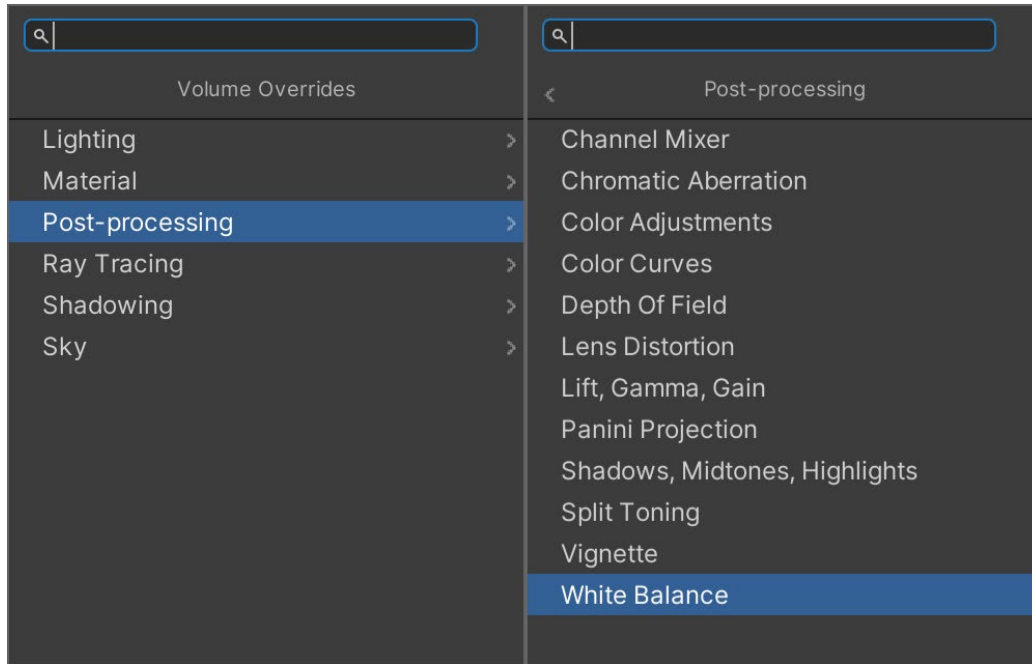


포스트 프로세싱 효과로 렌더링된 이미지를 더 영화처럼 만듭니다.

HDRP 포스트 프로세싱은 볼륨 시스템을 사용해 카메라에 이미지 효과를 적용합니다. 오버라이드를 어떻게 추가하는지 알고 있다면, 다양한 포스트 프로세싱 효과를 적용하는 프로세스에도 이미 적응한 것과 마찬가지로입니다.

포스트 프로세싱 오버라이드

색상과 콘트라스트를 조정하는 대부분의 포스트 프로세싱 오버라이드는 기능적으로 겹칠 수 있습니다. 여러 오버라이드를 적절하게 조합하려면 시행착오를 수반하기 마련입니다.



포스트 프로세싱 오버라이드

모든 효과가 필요하지는 않을 것입니다. 원하는 비주얼을 만드는 데 필요한 오버라이드만 추가하고 나머지는 무시하세요.

예시를 직접 사용해 보려면 Unity Hub에서 다운로드할 수 있는 HDRP 샘플 씬의 볼륨을 참고하세요.

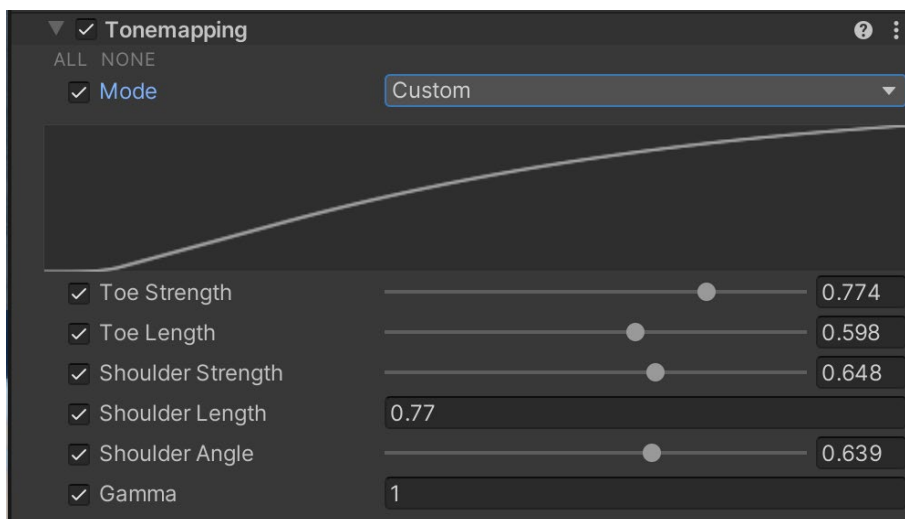
톤 매핑

톤 매핑은 **HDR** 색상을 HDR보다 제한된 화면의 **다이내믹 레인지**에 매핑하는 기술입니다. 톤 매핑을 사용하면 렌더링된 결과물의 콘트라스트와 디테일을 개선할 수 있습니다.



ACES 톤 매핑과 Neutral 톤 매핑 비교

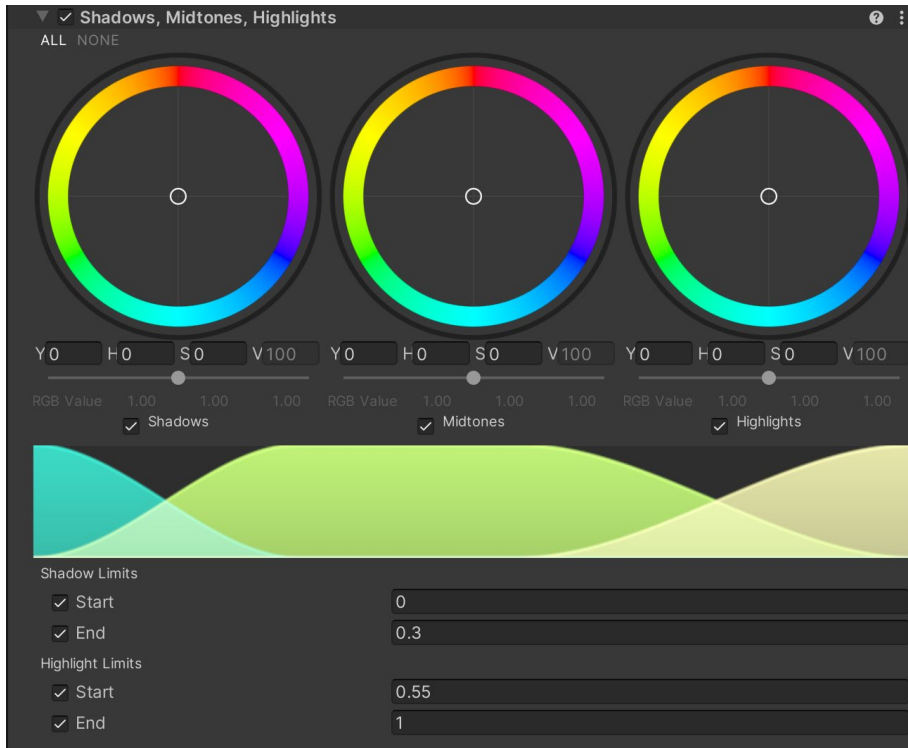
영화 같은 비주얼을 원한다면 **Mode**를 업계 표준인 **ACES(Academy Color Encoding System)**로 설정합니다. 채도와 콘트라스트가 낮은 비주얼을 원한다면 **Neutral**을 선택합니다. 숙련된 사용자라면 **Custom**을 선택하고 톤 매핑 곡선을 직접 정의할 수도 있습니다.



커스텀 커브를 사용한 톤 매핑

그림자, 미드톤, 하이라이트

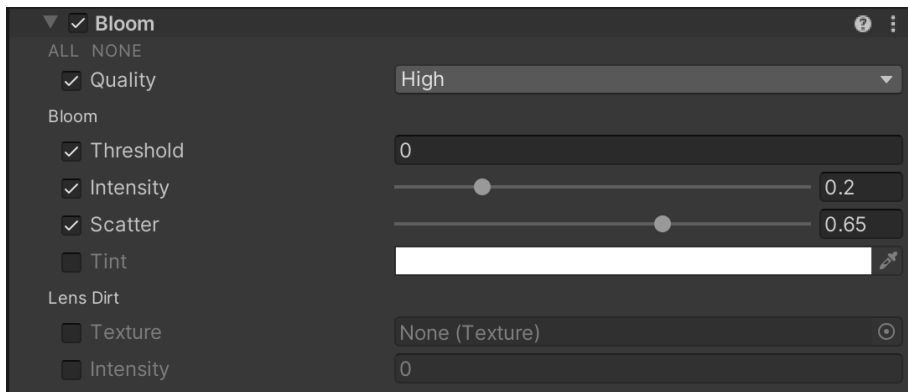
Shadows, Midtones, Highlights 오버라이드는 각각 렌더의 그림자, 중간 톤, 하이라이트에 대한 색조와 색상 범위를 제어합니다. 각 트랙볼을 활성화하여 이미지의 각 영역에 영향을 줄 수 있습니다. 그런 다음 **Shadow Limits**와 **Highlight Limits**를 사용하여 클리핑을 방지하거나 색상 보정이 너무 과해지지 않도록 합니다.



그림자, 미드톤, 하이라이트

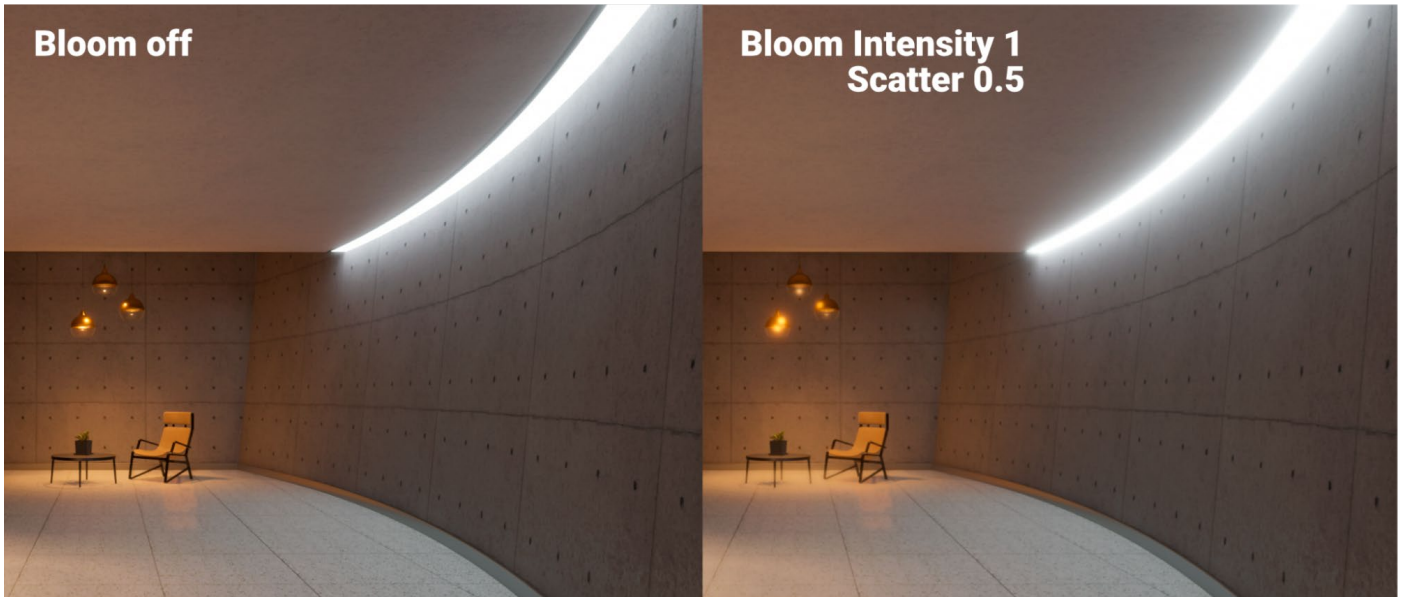
블룸

블룸은 광원 주변의 빛 번짐 효과를 생성합니다. 블룸을 사용하면 카메라를 메울 듯 강렬한 느낌의 광원을 연출할 수 있습니다.



Bloom 오버라이드

Intensity와 **Scatter**를 조정하여 블룸의 크기와 밝기를 조정할 수 있습니다. **Lens Dirt**는 번진 텍스처나 먼지 텍스처를 적용하여 블룸 효과를 회절시킵니다. **Threshold**를 사용하여 어두운 픽셀의 선명도를 유지합니다.



블룸 효과

덱스오브필드(피사계심도)

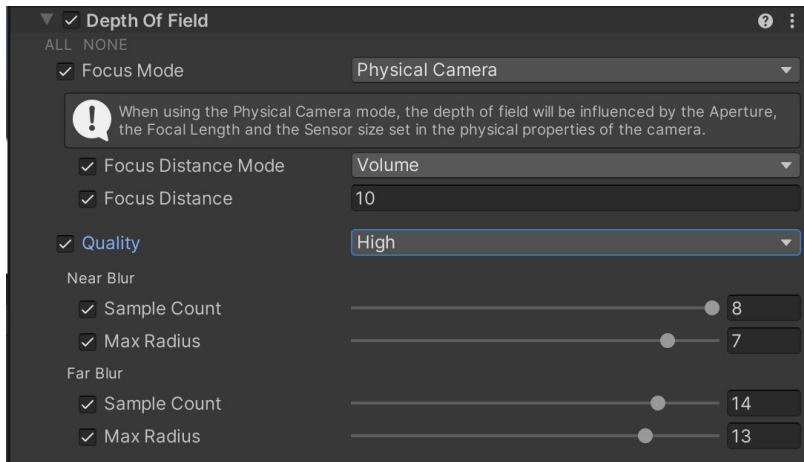
덱스오브필드는 실제 카메라 렌즈의 초점과 같은 특성을 시뮬레이션합니다. 카메라의 초점 거리에서 더 가깝거나 멀리 있는 오브젝트를 흐릿하게 만드는 효과입니다.

초점 거리를 설정할 수 있는 곳은 다음과 같습니다.

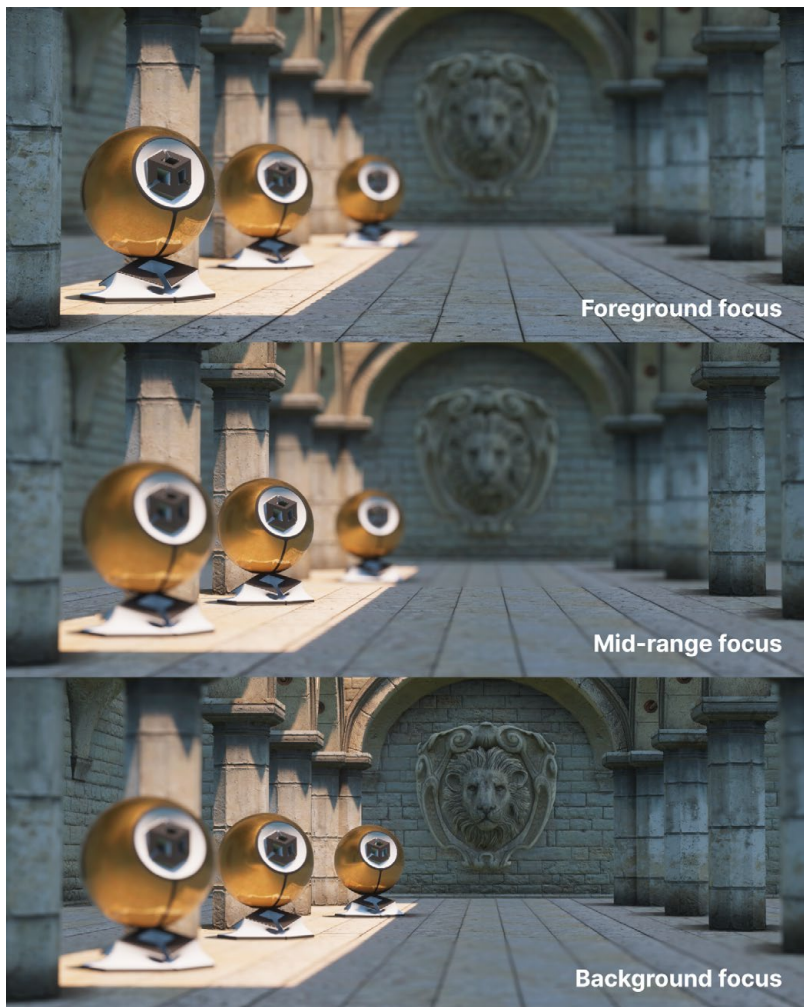
Focus Mode에 **Manual Ranges**가 설정된 볼륨 오버라이드: 여기서는 볼륨 자체가 초점 거리를 제어합니다. 예를 들어 오버라이드를 사용하여 위치(예: 수중 씬)에 따라 의도적으로 카메라에 블러 효과를 입힐 수 있습니다.

- [Volume Settings Extension](#)이 설정된 Cinemachine 카메라 사용: 이렇게 하면 대상을 따라가고 초점을 자동으로 맞출 수 있습니다.
- Focus Mode에 **Physical Camera**가 설정된 **Physical Camera** 프로퍼티: 이를 사용하면 Camera 컴포넌트에서 **Focus Distance** 파라미터를 애니메이션화할 수 있습니다.

Depth of Field를 활성화하면 이미지의 밝은 부분 주변에 **보케**라는 초점이 맞지 않는 블러 효과가 나타날 수 있습니다. 카메라의 조리개 모양을 수정하면 보케의 형상을 변경할 수 있습니다. 자세한 내용은 위의 더 많은 물리적 카메라 파라미터를 참조하시기 바랍니다.

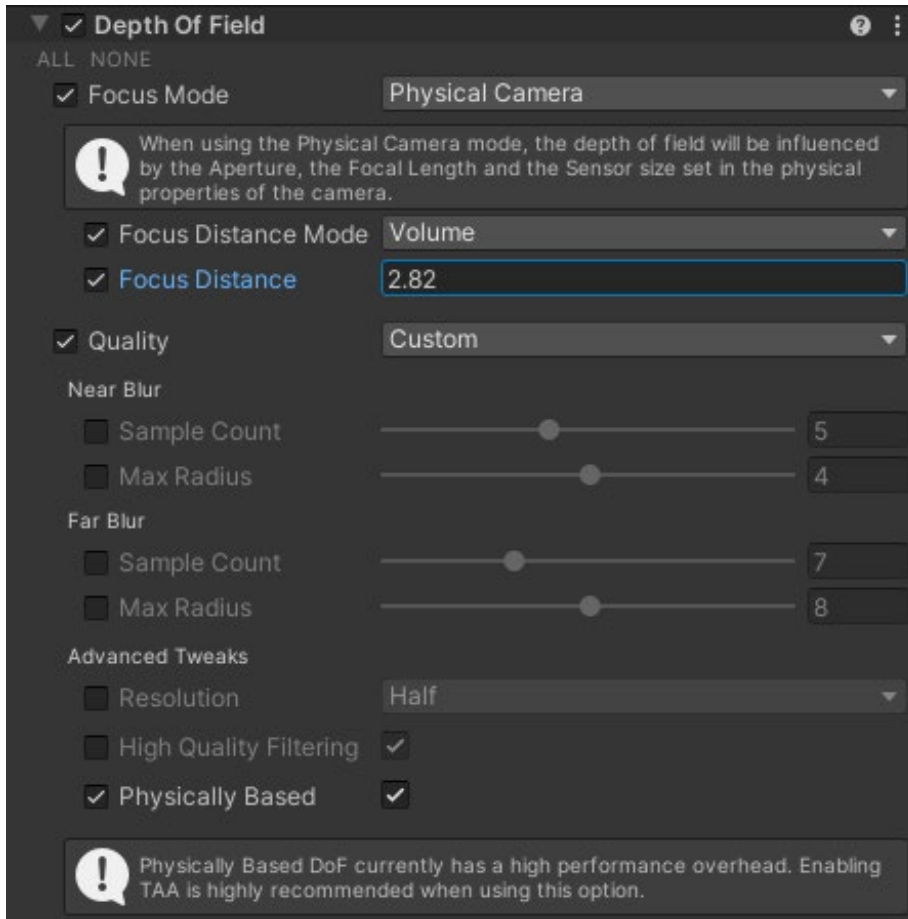


Depth of Field 오버라이드



덤소브필드로 실제 카메라의 초점 거리를 시뮬레이션합니다.

시네마틱 또는 오프라인 렌더링의 경우, 리소스를 조금 더 많이 소모하는 방법이지만 Additional Settings와 **Custom Quality**를 활성화해 **Physically Based**로 설정한 덤스오브필드를 사용해 볼 수 있습니다.



화이트 밸런스

White Balance 오버라이드는 최종 이미지에서 흰색이 올바르게 렌더링되도록 씬의 컬러를 조정합니다. **Temperature**를 선택해 노란색(따뜻한 색조)과 파란색(차가운 색조) 사이에서 값을 조정할 수 있습니다. **Tint**는 초록색과 마젠타색 사이의 색상 캐스트를 조정합니다.

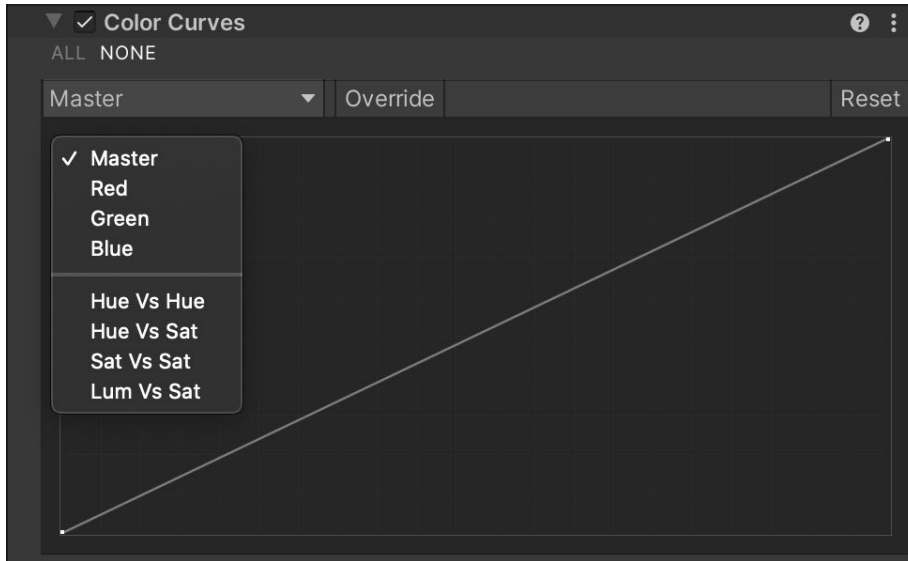
HDRP 샘플 프로젝트의 공간별 로컬 볼륨에는 White Balance 오버라이드가 포함되어 있습니다.



화이트 밸런스

색상 커브

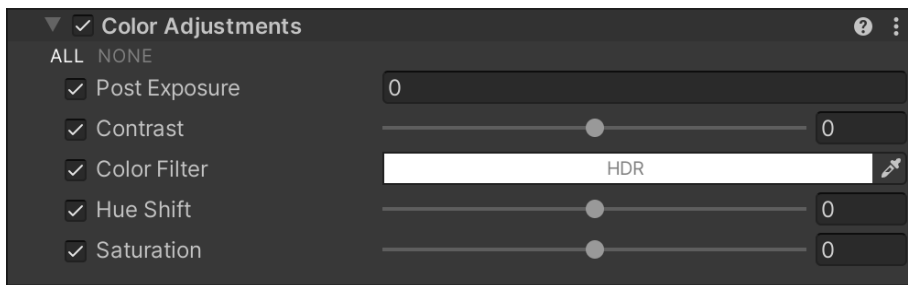
[그레이딩 커브](#)를 사용하면 색조, 채도, 광도에 대한 구체적인 범위를 조정할 수 있습니다. 사용할 수 있는 8개의 그래프 중 하나를 선택해 색상과 콘트라스트를 다시 매핑하세요.



색상 커브

색상 조정

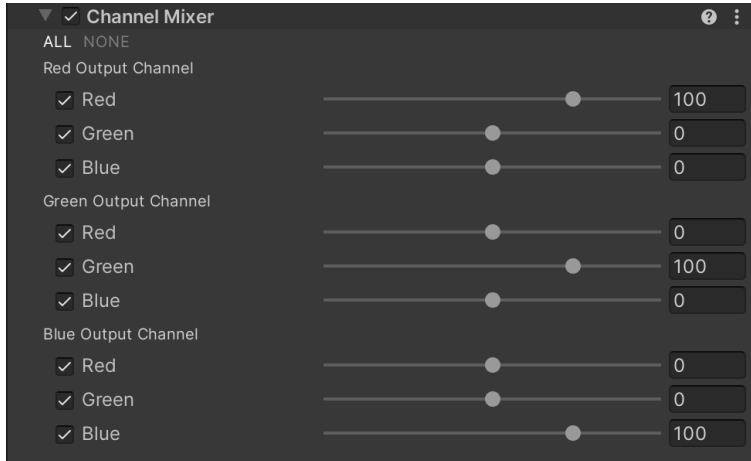
색상 조정 효과를 사용하면 최종 렌더링된 이미지의 전반적인 톤, 밝기, 색조, 콘트라스트를 조정할 수 있습니다.



색상 조정

채널 믹서

채널 믹서를 사용하면 한 컬러 채널이 다른 컬러 채널의 믹스에 영향을 줄 수 있습니다. RGB 출력을 선택한 다음, 입력 중 하나가 미치는 영향을 조정하면 됩니다. 예를 들어, Red Output Channel에서 Green 값을 높이면 이미지의 모든 초록색 영역이 붉은 빛을 띠게 됩니다.

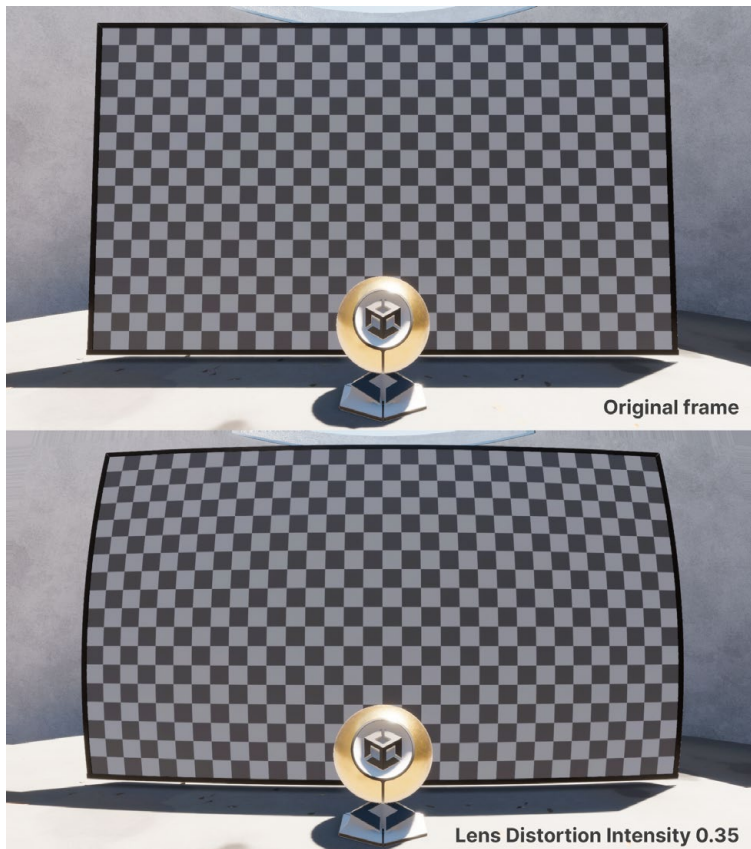


채널 믹서

렌즈 왜곡

렌즈 왜곡은 실제 렌즈 제조의 불완전성으로 인해 발생하는 방사형 패턴을 시뮬레이션합니다. 렌즈 왜곡을 사용하면 특히 줌 렌즈나 광각 렌즈에서 직선이 약간 휘거나 구부러져 나타납니다.

이 어안(fisheye) 효과를 사용하면 특별한 분위기나 스타일을 연출할 수 있습니다. 예를 들어 회상 장면 시퀀스를 강조하거나 캐릭터의 심경 변화를 나타낼 때 사용하기에 적합합니다. 또한 긴장감을 높이거나 방향 감각을 잃게 만드는 용도로도 좋습니다.



렌즈 왜곡은 방사형 패턴으로 이미지를 왜곡합니다.



비네트

비네트(Vignette)는 이미지의 가장자리가 어두워지거나 채도가 낮아지는 실제 사진의 효과를 모방합니다. 비네트 현상은 광각 렌즈를 사용하거나 빛을 막는 기기(렌즈 후드나 누적 필터 링)로 인해 발생할 수 있습니다. 비네트 효과는 보는 사람의 주의를 화면의 중앙으로 끌어들이는 데에도 사용할 수 있습니다.



비네트를 활용해 프레임 가장자리를 어둡게 만듭니다.

모션 블러

실제 오브젝트가 카메라의 노출 시간보다 빠르게 움직이면 결과 이미지에서는 오브젝트에 잔상이 생긴 것처럼 보이거나 흐릿하게 보입니다. Motion Blur 오버라이드는 이런 효과를 시뮬레이션합니다.

성능 리소스를 최소화하려면 Sample Count 값을 줄이고, Minimum Velocity 값은 높이고, Maximum Velocity 값은 줄입니다. Additional Properties에 있는 Camera Clamp Mode 파라미터를 줄여도 됩니다.

렌즈 플레어

렌즈 플레어는 카메라 렌즈에 밝은 빛을 비출 때 나타나는 현상입니다. 하나의 밝은 빛으로 나타날 수도 있고, 카메라의 조리개 모양과 일치하는 여러 색상의 다각형 도형으로 나타날 수도 있습니다.

실제로는 플레어가 의도치 않은 효과로 나타날 수도 있지만, 내러티브나 감각적인 연출에 유용하게 활용할 수 있습니다. 예를 들자면 강렬한 렌즈 플레어로 플레이어의 주의를 집중시키거나, 구성 또는 장면의 분위기를 바꾸는 데 사용하는 식입니다. 이 [Wikipedia 문서](#)에서 렌즈 플레어의 물리적인 측면을 더 자세히 알아볼 수 있습니다.



실제 사진의 렌즈 플레어입니다(출처: Wikipedia).

렌즈 플레어는 렌더링 프로세스 후반에 렌더링되므로 기본적으로는 포스트 프로세싱 효과입니다.

HDRP는 **Lens Flare (SRP)** 컴포넌트와 **Screen Space Lens Flare**(Unity 6에서 새로 도입) 두 가지 유형의 상호 보완적인 렌즈 플레어를 제공합니다.

Lens Flare (SRP) 컴포넌트

Lens Flare (SRP) 컴포넌트는 씬의 특정 오브젝트(주로 광원이나 밝은 요소)로부터 위치 기반 플레어를 생성합니다. 이를 사용하려면 오브젝트에 Lens Flare (SRP) 컴포넌트를 추가하고 [Lens Flare Data](#) 에셋을 할당하면 됩니다.

각 Lens Flare Data 에셋은 다각형, 원 또는 커스텀 이미지와 같은 여러 형태의 요소들로 구성되며, 이러한 요소를 레이어링하여 복잡한 효과를 만들 수 있습니다. 광원에 연결하면 광원의 색상을 상속받기 때문에 하나의 Flare 에셋을 여러 광원에서 재사용할 수 있습니다.

Lens Flare (SRP)는 강도, 스케일, 오클루전 파라미터를 제어합니다.



Lens Flare 샘플은 사실적인 렌즈 효과부터 스타일라이즈드에 가까운 효과까지 다양한 프리셋을 제공합니다.

스크린 공간 렌즈 플레어

Unity 6에는 새로운 포스트 프로세싱 기법인 **Screen Space Lens Flare 오버라이드(SSLF)**가 추가되었습니다. 이 기법을 사용하면 밝은 스페큘러 하이라이트나 발광 메시와 같은 모든 밝은 표면에서 플레어를 생성할 수 있습니다.

SSLF는 볼륨 오버라이드(**Post-processing > Screen Space Lens Flare**)로 사용할 수 있습니다. 이 유형의 플레어 효과는 발광 표면, 스페큘러 반사, 화면에 보이는 광원처럼 씬에서 밝게 보이는 영역에 작용합니다. HDRP는 이러한 밝은 영역을 추출한 다음 스트레치, 블러, 색 수차와 같은 효과를 적용하여 플레어를 생성합니다.



스크린 공간 렌즈 플레어는 블룸 효과와 동일한 버퍼를 사용하여 여러 종류의 플레어를 생성합니다.

- Regular 플레어는 화면의 밝은 영역에서 나온 왜곡된 하이라이트입니다.
- Reversed 플레어는 Regular 플레어를 상하 반전하고 뒤집은 것입니다.
- Warped 플레어는 Regular 플레어를 극좌표 기반으로 변환하여 원형 카메라 렌즈를 모방한 것입니다.
- Streaks는 아나모픽 카메라 렌즈를 모방하도록 한 방향으로 길게 늘린 플레어입니다.

Screen Space Lens Flare를 활성화하려면 먼저 HDRP 에셋과 Frame Settings에서 활성화한 다음, Volume에 **Screen Space Lens Flare** 오버라이드를 추가하고 Intensity를 0보다 크게 설정합니다.

이미 Bloom 오버라이드가 활성 상태인 경우 Screen Space Lens Flare가 보이려면 Bloom의 Intensity가 0보다 커야 합니다.

Screen Space Lens Flare와 Lens Flare (SRP) 컴포넌트를 함께 사용해 더 세밀하게 제어할 수도 있습니다. 두 시스템 모두 포스트 프로세싱 효과로 렌더링되며, 렌즈 플레어의 오클루전은 구름, 볼류메트릭 안개, 물과 같은 환경 요소와 통합됩니다.



스크린 공간 렌즈 플레어는 선형 플레어 효과를 더할 수 있습니다.

시작하기

렌즈 플레어를 직접 추가하는 가장 좋은 방법은 패키지 관리자에서 High Definition RP의 샘플을 설치하는 것입니다. 샘플을 설치하면 미리 정의된 플레어 에셋이 프로젝트에 추가됩니다. 샘플에는 렌즈 플레어를 살펴보고 직접 만들어 볼 수 있는 여러 테스트 씬도 포함되어 있습니다.

미리 정의된 Flare Data 에셋은 일반적으로 사용되는 여러 플레어 효과를 재현합니다. 각 렌즈 플레어가 어떻게 구성되었는지 자세히 알아보려면 **Directional Light Sun** 씬과 **Point Light** 씬을 살펴보세요.

- 각 광원에는 자체 Lens Flare (SRP) 컴포넌트가 있습니다.
- 이 컴포넌트는 프로젝트의 Lens Flare Data 에셋을 참조합니다.

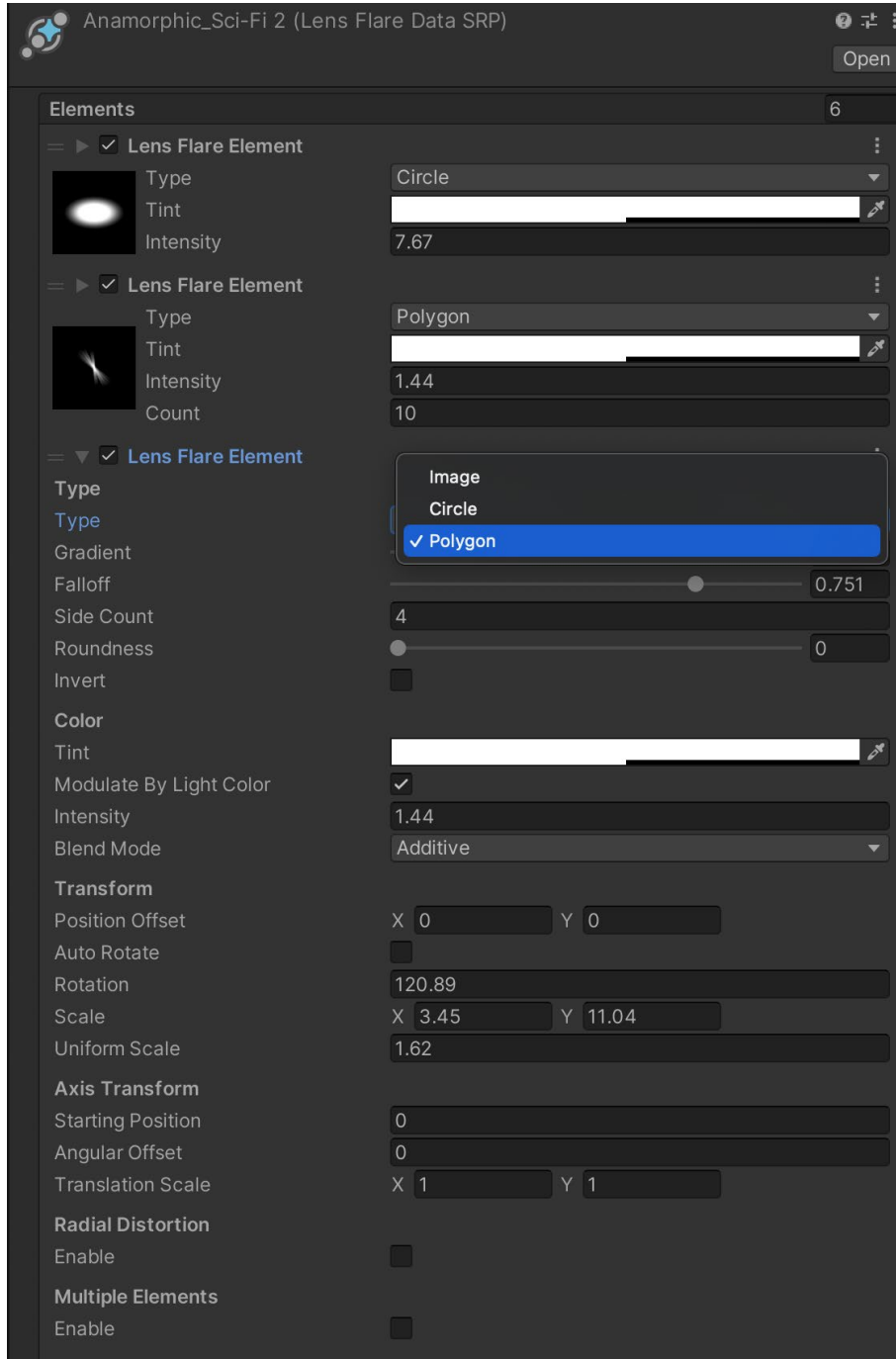
Lens Flares 씬을 살펴보면 사실적인 렌즈 효과부터 스타일라이즈드에 가까운 효과까지 다양한 프리셋을 바탕으로 자체 플레어를 만드는 데 도움을 받아 보세요.

프로젝트에서 **Directional Light Sun**을 선택하면 광원과 연결된 Lens Flare (SRP) 컴포넌트와 렌즈 플레어 데이터 에셋을 찾을 수 있습니다. 에셋을 변경하며 다양한 플레어 효과를 살펴보세요.



여러 렌즈 플레어를 함께 사용하여 원하는 효과를 얻습니다(예시에서는 강도가 증가됨).

렌즈 플레어 작동 방식은 [이 SIGGRAPH 발표](#)에서도 자세히 다루고 있습니다.



커스텀 플레어 효과 라이브러리를 만들 수도 있습니다.

다이내믹 해상도

업스케일링은 게임을 낮은 해상도로 렌더링한 뒤 더 높은 해상도로 스케일을 높여 성능을 향상시키는 기법입니다. 이를 통해 시각적 품질을 유지하면서 GPU 워크로드를 줄여, 디테일의 큰 손실 없이 더 높은 프레임 속도를 얻을 수 있습니다.

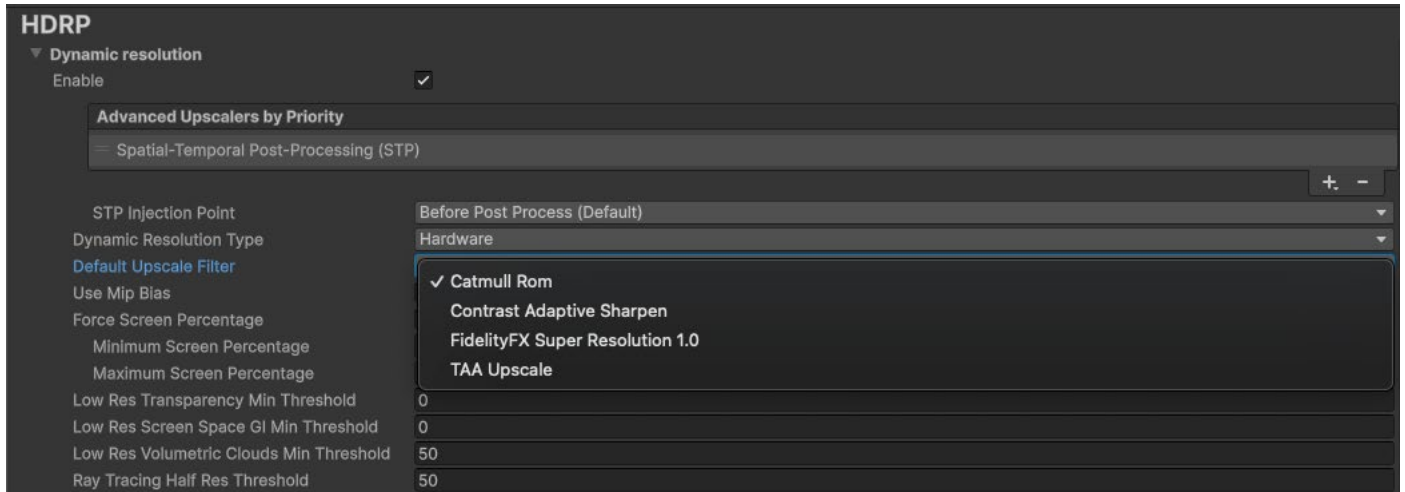
다이내믹 해상도는 GPU 성능에 따라 렌더링 해상도를 실시간으로 조절하는 HDRP의 기능입니다. 다이내믹 해상도가 활성화되면 HDRP는 가능한 한 최대의 해상도를 기준으로 렌더 타겟을 할당하고 뷰포트를 다시 스케일링하여 다양한 해상도로 렌더링합니다. 각 프레임 끝에서 HDRP는 렌더링된 이미지를 디스플레이의 기본 해상도에 맞게 업스케일링합니다.

HDRP는 하드웨어 및 소프트웨어 다이내믹 해상도를 모두 지원합니다. 하드웨어 다이내믹 해상도는 콘솔이나 DirectX 12, Vulkan, Metal을 사용하는 PC에서 사용할 수 있습니다. 플랫폼이 하드웨어 다이내믹 해상도를 지원하지 않는 경우 HDRP는 소프트웨어 다이내믹 해상도를 사용합니다.

HDRP 에셋의 **Rendering > Dynamic Resolution**에서 다이내믹 해상도를 활성화한 다음, Camera의 **Rendering** 설정에서 **HDRP Dynamic Resolution**을 선택합니다.

HDRP는 이제 다양한 하드웨어에서 업스케일링 동작을 더 세부적으로 제어하는 옵션을 제공합니다.

- **Advanced Upscalers by Priority:** 이 옵션을 사용하면 DLSS, FSR, STP와 같은 고급 업스케일러의 우선순위를 지정할 수 있습니다. HDRP는 플랫폼에서 지원되는 가장 높은 우선순위의 업스케일러를 선택합니다. 사용 가능한 업스케일러가 없으면 **Default Upscale Filter**로 폴백합니다.
- **Default Upscale Filter:** 사용 가능한 고급 업스케일러가 없는 경우의 폴백 업스케일링 방법(예: TAAU, Catmull-Rom, CAS)을 설정합니다.



새 사용자 인터페이스에서 업스케일링을 위한 폴백 환경 설정을 지정합니다.

Unity HDRP는 다양한 업스케일 필터 방법을 제공합니다.

Catmull-Rom

Catmull-Rom은 4개의 쌍선형 샘플을 사용합니다. 가장 적은 리소스를 사용하지만, HDRP가 업스케일링 단계를 수행한 후 이미지가 흐릿해질 수 있습니다. Catmull-Rom은 종속성이 없으며 포스트 프로세싱 파이프라인의 끝에서 실행됩니다.

CAS

CAS(Contrast Adaptive Sharpen)는 AMD의 FidelityFX(CAS)를 사용합니다. 이 방법을 사용하면 강한 샤프닝 단계를 통해 선명한 이미지를 생성할 수 있습니다. 다이내믹 해상도 화면 비율이 50% 미만인 경우에는 이 옵션을 사용하지 마세요. AMD FSR(FidelityFX™ Super Resolution)과 Contrast Adaptive Sharpening에 대한 자세한 내용은 [AMD FSR\(FidelityFX Super Resolution\)](#)을 참고하세요.

CAS(Contrast Adaptive Sharpen)는 종속성이 없으며 포스트 프로세싱 파이프라인의 끝에서 실행됩니다.

NVIDIA DLSS(NVIDIA RTX GPU 및 Windows용)

NVIDIA DLSS는 Unity 6의 HDRP에서 기본적으로 지원됩니다. NVIDIA DLSS는 AI를 사용하여 추가 프레임을 생성하고 기본 해상도에 가까운 품질의 이미지를 구성하면서도 실제로는 극히 일부 픽셀만 기존 방식으로 렌더링하는 신경망 기반 렌더링 기술 모음입니다.

실시간 레이트레이싱과 NVIDIA DLSS를 사용하면 NVIDIA RTX GPU에서 더 높은 프레임 속도와 해상도로 구동되는 사실적인 월드를 제작할 수 있습니다. 또한 DLSS는 기존 래스터화 기반 그래픽스의 성능을 크게 개선합니다. [NVIDIA의 Unity 개발자 페이지](#)(영문)와 [Unity 기술 자료](#)에서 더 자세한 내용을 알아보세요.



DLSS 기술을 통해 24 Entertainment의 나라카: 블레이드포인트와 같은 게임을 4K로 실행할 수 있습니다.

AMD FSR(크로스 플랫폼)

AMD FSR(FidelityFX™ Super Resolution)에는 빌트인 HDRP 지원이 탑재되어 있습니다. HDRP 에셋과 카메라에서 다이내믹 해상도를 활성화하고 Upscale 필터 옵션에서 **FidelityFX Super Resolution 2.0**을 선택하면 FSR을 사용할 수 있습니다.

AMD FSR(FidelityFX Super Resolution)은 저해상도 입력으로부터 고해상도 프레임을 생성하는 뛰어난 오픈 소스 솔루션입니다. FSR은 고품질의 에지를 만드는 데 주력하는 알고리즘을 사용하여 기본 해상도에서 직접 렌더링하는 것보다 성능을 개선합니다.

또한 하드웨어 레이트레이싱처럼 리소스 소모가 많은 렌더 작업 등에서 실질적인 성능을 개선합니다. 더 자세한 내용은 [AMD 웹페이지](#)와 [Unity 포럼](#)에서 확인할 수 있습니다.



[Steam](#)에서 다운로드할 수 있는 스페이스쉽 데모로 FSR 기술을 체험해 보세요.

TAA Upscale(크로스 플랫폼)

TAA(Temporal Anti-Aliasing) Upscale은 시간적 통합을 사용하여 이미지 선명도를 향상하며, HDRP의 일반 안티앨리어싱과 함께 동작합니다. HDRP는 포스트 프로세싱 전에 TAA와 동시에 업스케일 필터를 적용합니다.

TAA Upscale은 다른 안티앨리어싱 방법과 호환되지 않으므로 이 기능을 활성화할 때는 TAA를 사용해야 합니다. 다이내믹 해상도가 활성화되어 있으면 화면 해상도가 100%인 경우에도 계속 작동합니다. 업스케일 방법은 HDRP 에셋에서 설정하거나 코드를 통해 동적으로 조정할 수 있습니다.

자세한 내용은 기술 자료의 [TAA Upscale에 대한 참고 사항](#) 섹션을 참조하세요.



TAA Upscale이 사용된 왼쪽 이미지(A)는 더 선명하고 뚜렷해 보입니다. Catmull-Rom 업스케일 방법이 사용된 오른쪽의 이미지(B)는 더 부드럽고 덜 뚜렷해 보입니다.

시공간 포스트 프로세싱(크로스 플랫폼)

STP(시공간 포스트 프로세싱)는 Unity 6의 새로운 빌트인 업스케일러입니다. STP는 낮은 해상도로 렌더링한 뒤 공간 및 시간 데이터를 사용해 고해상도 이미지를 다시 제작합니다. 이로 인해 비주얼 충실도는 그대로 유지되면서 GPU 워크로드가 줄어듭니다.

STP는 공간·데이터와 시간 데이터를 둘 다 사용하여 계단 현상을 줄이는 안티앨리어싱 기법을 사용합니다. 공간 데이터로는 단일 프레임 내의 디테일을 선명하게 정제하고, 시간 데이터로는 여러 프레임에 걸쳐 모션을 안정화합니다. 결과적으로 STP는 더 균일하고 아티팩트가 적은 깨끗한 비주얼을 제공합니다.



STP는 이미지 품질을 유지하면서 GPU 워크로드를 줄여 줍니다.

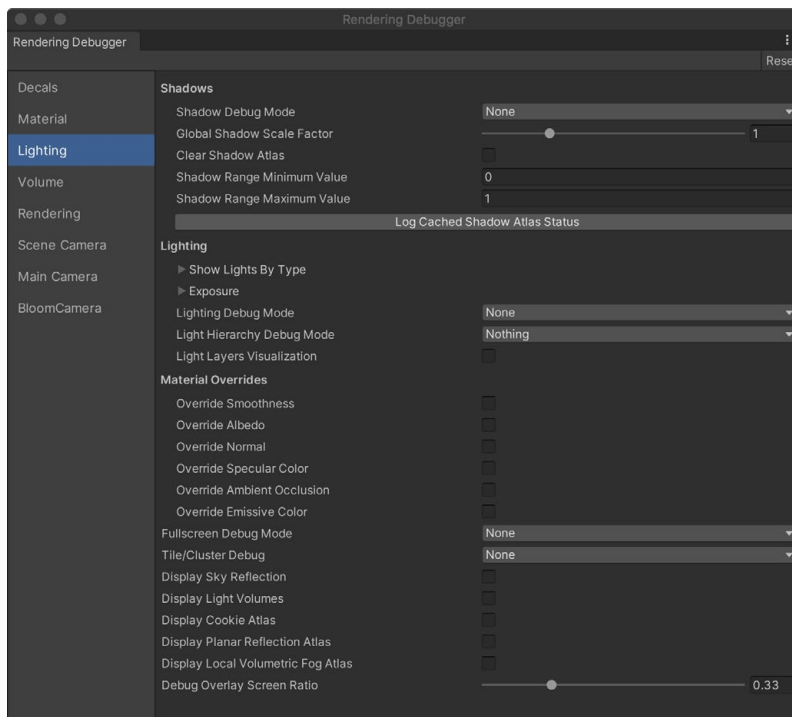
STP는 플랫폼에 따라 설정을 자동으로 조정합니다. 고성능 기기(PC 및 콘솔)에서는 업스케일링 중에 더 높은 품질의 이미지 필터링과 디링잉(deringing) 로직을 사용합니다. 모바일 기기에서는 성능 영향을 최소화하기 위해 더 빠른 필터링 방법을 사용하면서도 DLSS2, FSR2, XeSS와 비슷한 비주얼 품질을 제공합니다.

HDRP 에셋의 Render Scale을 조정하면 타겟 플랫폼에 따라 STP의 성능과 품질의 균형을 미세 조정할 수 있습니다.

STP는 Shader Model 5.0을 필요로 하며 TAA(Temporal Anti-Aliasing)와만 호환됩니다. 또한 하드웨어 지원이 제공되지 않는 경우 다이내믹 해상도를 지원하지 않습니다. 하드웨어 다이내믹 해상도를 사용할 수 없는 경우 Render Scale을 고정 값으로 설정해야 합니다.

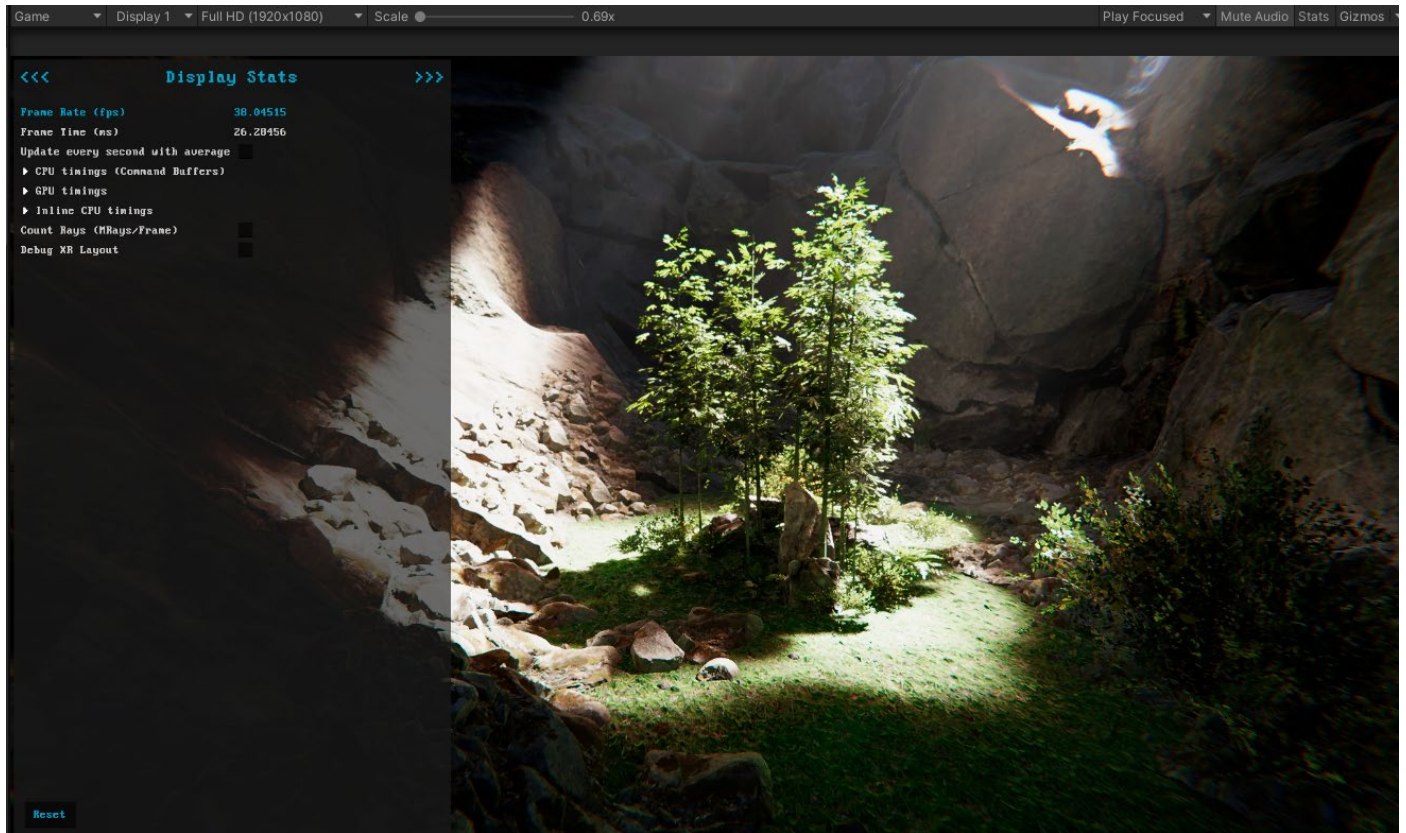
렌더링 디버거

Rendering Debugger 창(**Window > Analysis > Rendering Debugger**)에는 스크립터블 렌더 파이프라인 전용 디버깅 툴과 시각화 툴이 있습니다. 왼쪽에는 카테고리가 정리되어 있으며, 각 패널을 사용하면 조명, 머티리얼, 볼륨, 카메라 등과 관련된 문제를 따로 확인할 수 있습니다.



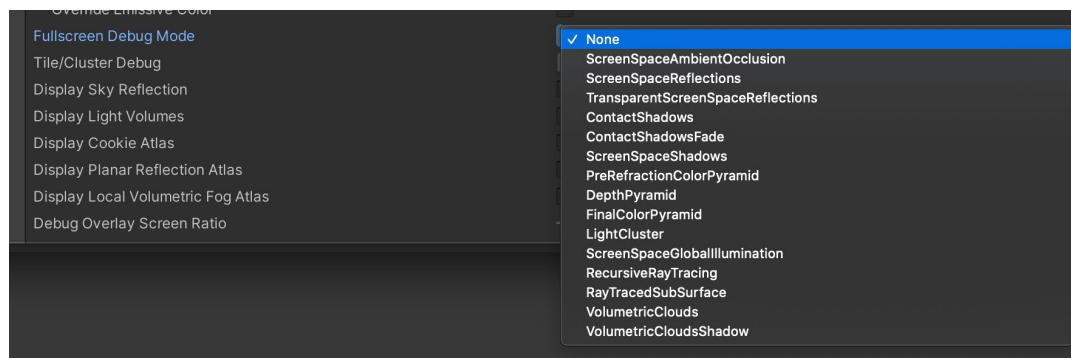
렌더링 디버거

렌더링 디버거는 런타임 중 플레이 모드의 게임 뷰에서도 사용할 수 있으며, 개발 빌드로 빌드한 플레이어에서도 사용할 수 있습니다. **Ctrl+Backspace**를 누르거나 게임 컨트롤러의 스틱 두 개를 동시에 누르면 메뉴가 열립니다.



게임 뷰 또는 플레이어 버전의 Rendering Debugger 창

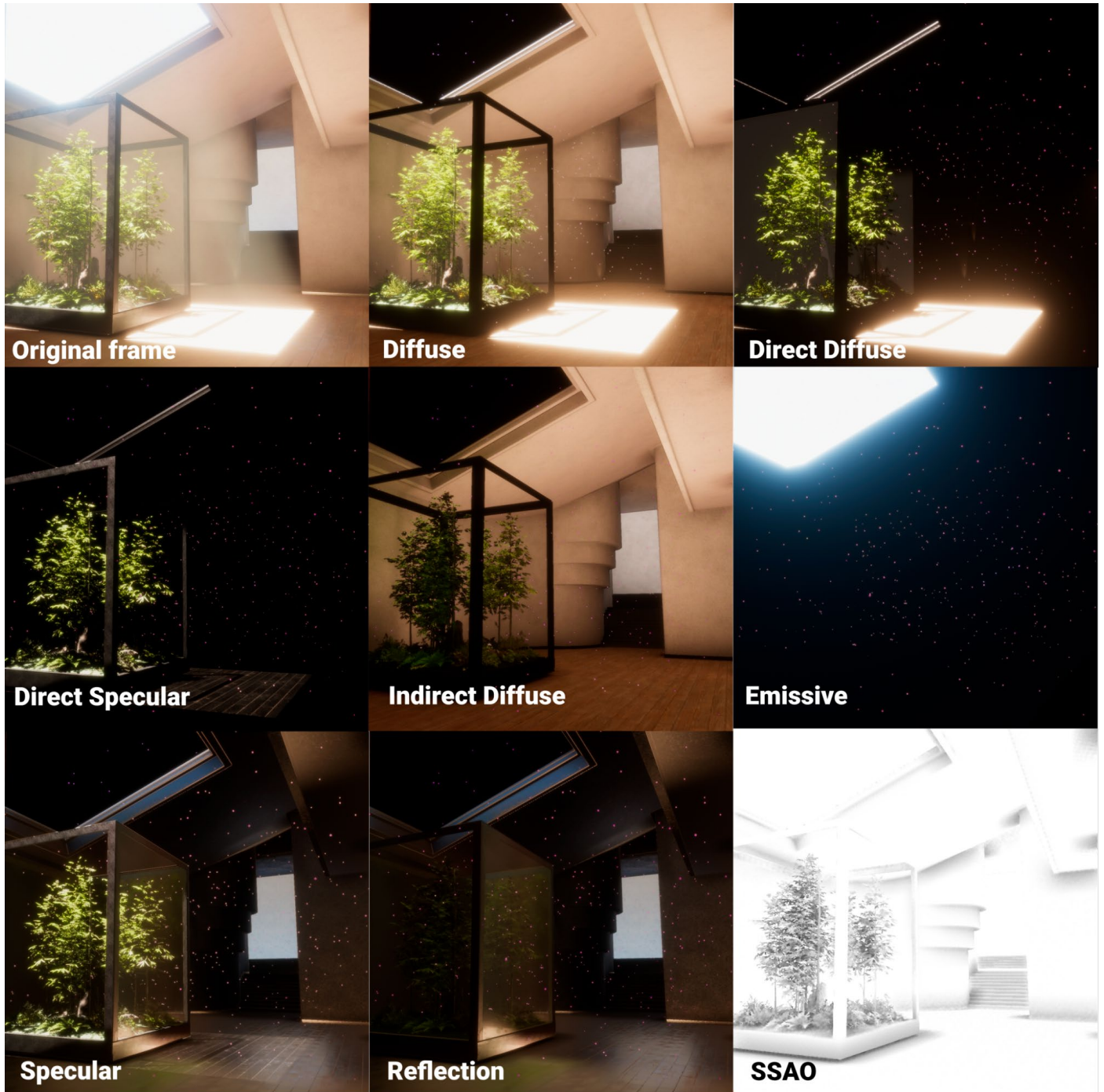
디버거를 사용하면 특정 렌더링 패스의 문제를 해결할 수 있습니다. Lighting 패널에서 **Fullscreen Debug Mode**로 이동해 디버깅할 기능을 선택할 수 있습니다.



Fullscreen Debug Mode 옵션

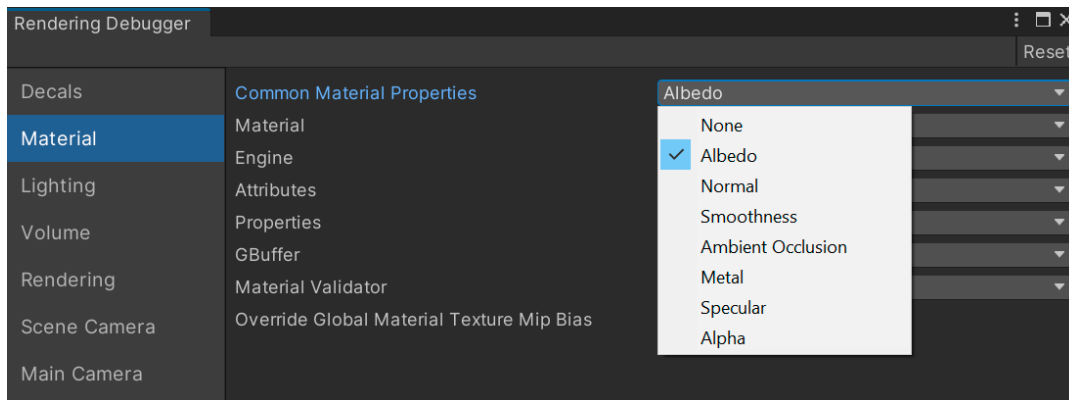
디버그 모드를 사용하면 픽셀을 확인하고 특정 조명 문제나 셰이딩 문제가 발생하는 원인을 파악할 수 있습니다. 왼쪽에 있는 패널은 카메라, 머티리얼, 볼륨 등 중요한 설정과 값을 표시하며, 렌더링을 최적화하는 데 활용할 수 있습니다.

Fullscreen Debug Mode를 활성화하면 씬 뷰와 게임 뷰가 일시적으로 특정 기능을 시각화하도록 전환되므로, 문제를 파악하는 데 유용하게 활용할 수 있습니다.

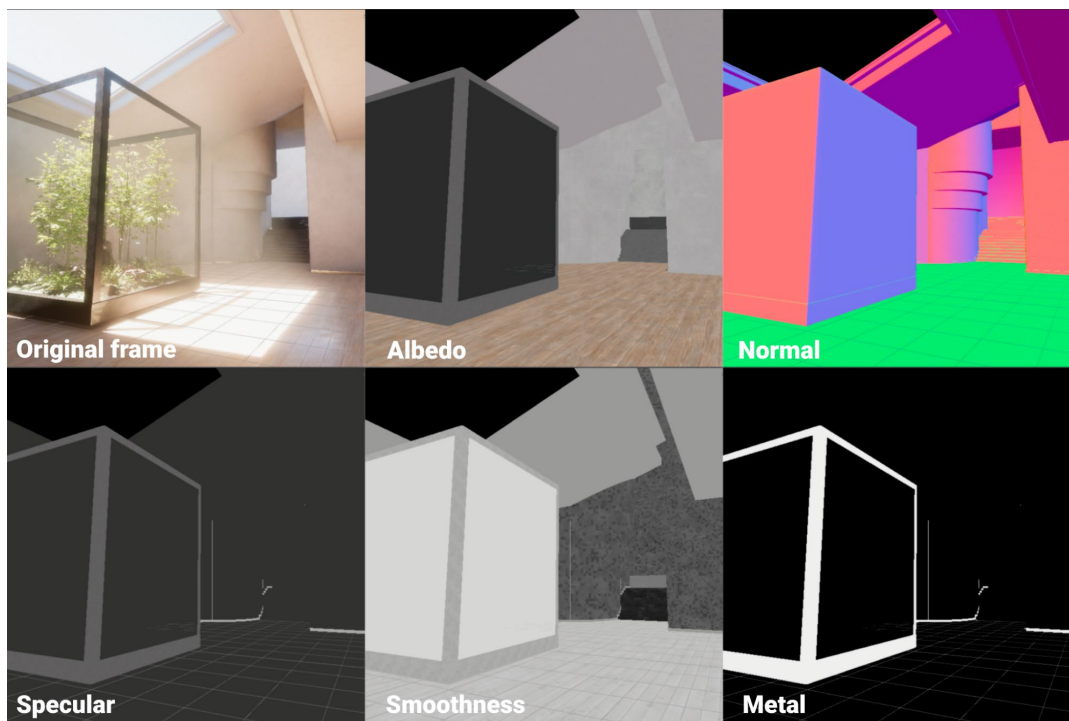


Lighting Debug Mode 또는 Fullscreen Debug Mode로 씬의 광원을 파악합니다.

여러 일반 머티리얼 프로퍼티를 디버깅할 수도 있습니다. Material 패널의 Common Material Properties에서 Albedo, Normal, Smoothness, Specular 등을 선택하면 됩니다.



Common Material Properties



렌더 파이프라인 디버거를 사용해 머티리얼 문제를 해결합니다.

색상 모니터

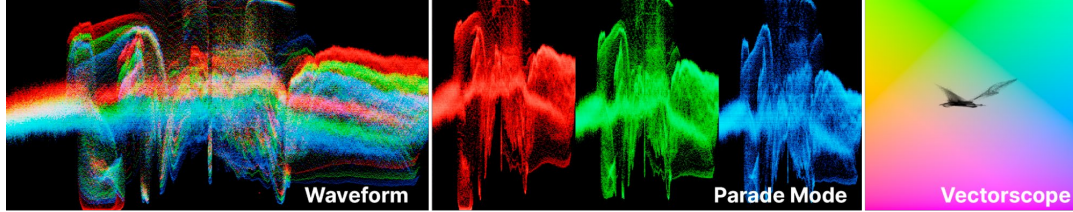
Unity에는 씬의 전반적인 모습과 노출을 제어할 수 있는 업계 표준 색상 모니터 세트가 포함되어 있습니다. Rendering 탭의 **Rendering Debugger** 창(Window > Analysis > **Rendering Debugger**)에서 이러한 모니터를 확인할 수 있습니다.

Waveform 모니터는 렌더링된 이미지의 휘도(밝기) 값을 그래픽으로 나타냅니다.

Parade Mode는 파형 이미지를 RGB(빨간색, 초록색, 파란색) 채널로 분할하므로 색상 불균형이나 불일치를 쉽게 파악하고 교정할 수 있습니다.

Vectorscope 모니터는 이미지의 색조(색상)와 채도(색상 강도)를 측정하는 원형 그래프를 제공합니다. 이 툴은 정확한 피부 톤을 구현하고 주로 쓰는 색상 캐스트를 식별하는 데 특히 유용합니다.

이러한 모니터를 워크플로에 통합하면 정확하고 균형 잡힌 색상을 구현할 수 있습니다.



색상 모니터

Runtime Frame Stats

SRP에서는 프로파일러를 통해 Runtime Frame Stats 패널에 액세스할 수 있습니다. 이 툴은 에디터에서만 아니라 플레이어에서도 사용할 수 있어 GPU 시간이 주로 어디에 사용되는지 손쉽게 확인할 수 있습니다.

아티스트, 테크니컬 아티스트, 개발자 등이 HDRP의 프레임별 성능을 자세히 파악하는 데 도움이 됩니다.

참고: 보다 정확한 파악을 위해 에디터가 아닌 플레이어 빌드에서 프로파일링하는 것이 좋습니다.

Player Settings에서 **Frame Timing Stats**를 활성화합니다. 이렇게 하면 플레이 모드에서 **Rendering Debugger**(에디터의 경우 **Window > Analysis > Rendering Debugger**, 플레이어의 경우 **Ctrl + Backspace**)를 통해 **Display Stats** 화면에 액세스할 수 있습니다.

Display Stats		Avg	Min	Max
Decals	Frame Rate (FPS)	59.1	43.6	76.9
Material	Frame Time	17.17ms	13.00ms	22.92ms
Lighting	CPU Main Thread Frame	8.75ms	7.34ms	12.17ms
Rendering	CPU Render Thread Frame	1.93ms	1.41ms	3.44ms
MainCamera	CPU Present Wait	0.00ms	0.00ms	0.00ms
Volume	GPU Frame	0.08ms	0.07ms	0.09ms
	▶ Bottlenecks			
	Update every second with average			
	▼ Detailed Stats	CPU	CPUInline	GPU
	HDRRenderPipelineAllRenderRequest	0.00ms	6.52ms	0.00ms
	VolumeUpdate	0.00ms	0.17ms	0.00ms
	RenderShadowMaps	1.02ms	0.05ms	0.23ms
	GBuffer	1.33ms	0.01ms	0.86ms
	PrepareLightsForGPU	0.03ms	0.74ms	0.00ms
	VolumeVoxelization	0.00ms	0.00ms	0.00ms
	VolumetricLighting	0.00ms	0.00ms	0.00ms

플레이 모드에서 확인할 수 있는 Runtime Frame Stats

더 자세한 내용은 [렌더 파이프라인 디버거](#) 기술 자료를 참조하세요.

HDR 디스플레이 지원

일반적인 다이내믹 레인지 디스플레이와는 다르게 HDR(High Dynamic Range) 디스플레이는 더 넓은 스펙트럼의 휘도 레벨을 재현하므로, 사람의 눈이 자연 환경을 인지하는 것에 가깝게 표현할 수 있습니다. Unity 6는 이제 에디터와 스탠드얼론 플레이어 양쪽에서 [크로스 플랫폼 HDR 출력 지원](#)을 제공합니다.

HDR 디스플레이가 점점 보편화됨에 따라 Unity는 이러한 디스플레이의 장점을 활용하여 더 높은 대비(하이라이트/그림자)와 색 채도를 갖춘 이미지를 재현할 수 있습니다.

HDRP는 HDR 이미지 렌더링을 오랫동안 지원해 왔지만, 이전에는 HDR 디스플레이 출력 전용 톤 매핑이 없었습니다. 출력이 최적화되지 않아 이미지의 가장 밝은 부분과 어두운 부분의 디테일이 손실될 우려가 있었습니다.

Unity 6는 모든 렌더 파이프라인과 지원되는 플랫폼으로 HDR 디스플레이 지원을 확장함으로써 네이티브 색 공간 렌더링과 HDR 톤 매핑을 가능하게 합니다. HDR 출력은 10비트 또는 16비트 색심도를 지원하며, 최대 680억 색상 범위를 제공합니다.

에디터와 스탠드얼론 플레이어는 이제 모든 SRP 렌더 파이프라인 및 호환 플랫폼에서 전체 HDR 톤 매핑과 디스플레이 지원을 제공합니다. HDR 렌더링은 더 넓은 색상 값을 저장하기 위해 부동 소수점 프레임버퍼를 사용합니다. 이는 SDR 렌더에 비해 조명, 포스트 프로세싱, 전체적인 비주얼 정확도를 향상합니다.

프로젝트에 HDR을 구현한 후에는 새로운 HDR 보정 템플릿을 사용하여 HDR 보정 메뉴를 구현할 수 있습니다. 이렇게 하면 과소/과다 노출과 같이 자주 발생하는 문제를 방지하고 화면의 텍스트 가독성을 높일 수 있습니다.



HDR 출력을 활성화하려면 **Project Settings > Player > Other Settings**로 이동하여 다음 설정을 활성화하세요.

— **Allow HDR Display Output**

— **Use HDR Display Output**

Use HDR Display Output은 기본 디스플레이에서 HDR 출력을 사용해야 하는 경우에만 활성화하세요.

HDR 출력은 게임 뷰와 플레이어에서만 활성화됩니다. 현재 이 기능은 DirectX 12에서만 사용할 수 있습니다.

HDR을 활성화하면 향상된 품질의 URP 렌더 출력 이미지를 HDR 디스플레이에 표시하는 옵션을 사용할 수 있습니다. 그 결과 해당 기기에서 자연광 환경을 더욱 효과적으로 모사하는 색상과 콘트라스트로 최종 이미지를 구현할 수 있습니다.

Unity 2022에서 추가되었던 기존 데스크톱 및 콘솔 지원에 더해 Unity 6에서는 다음 플랫폼에 대한 모바일 지원이 추가되었습니다.

— iOS 플레이어(iOS 16+, iPadOS 16+)

— Vulkan 및 GLES를 사용하는 Android 플레이어(Android 9+, 기기 성능에 따라 상이)

HDR 디스플레이를 지원하는 일반적인 모바일 기기로는 iPhone X 이상, 삼성 갤럭시 S10 이상, 갤럭시 노트 10 이상, 갤럭시 탭 S6 이상 버전이 있습니다.

자세한 내용은 [이 블로그 게시물](#)과 [HDR 출력](#)을 참조하세요.

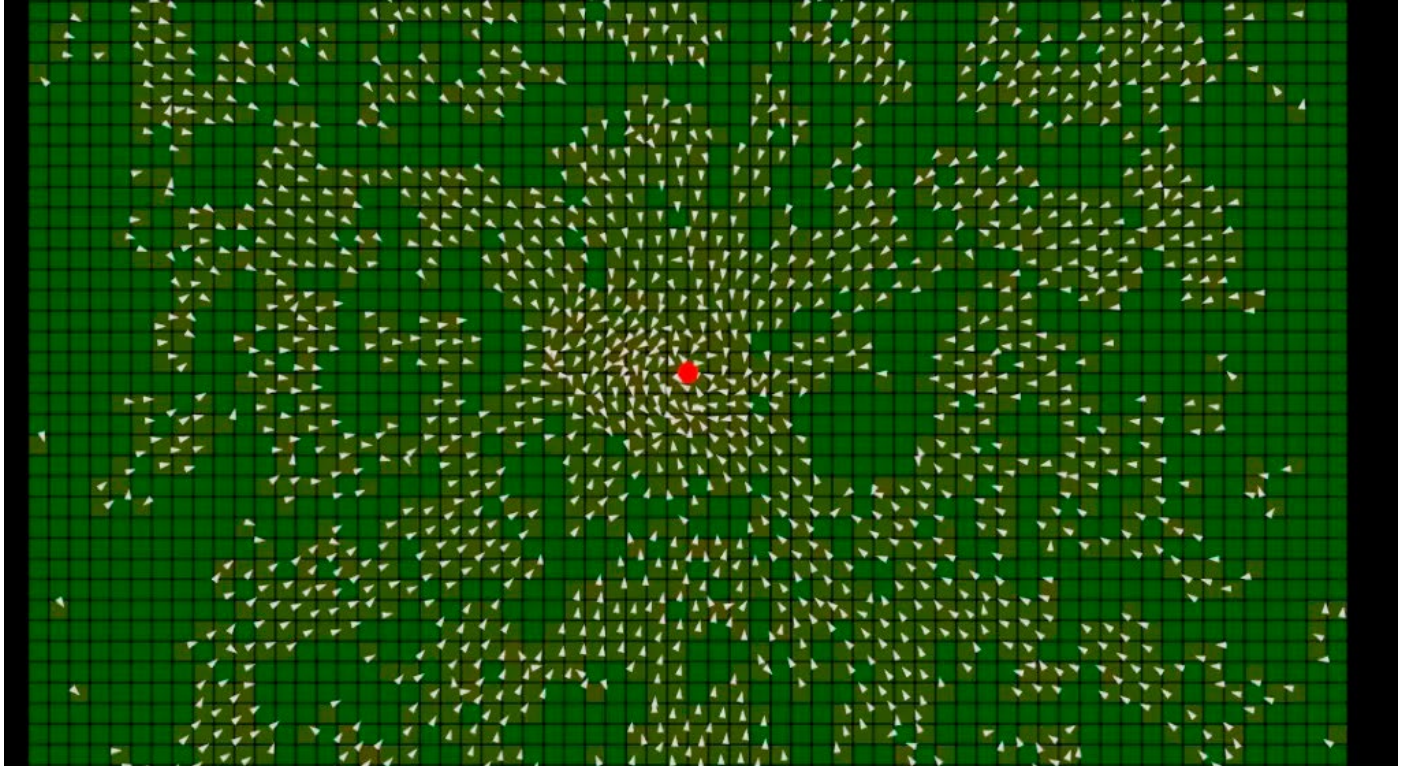
VFX Graph 지원

Unity 6에는 HDRP와 함께 작동하는 VFX Graph 기능도 포함되어 있어 더욱 유연한 워크플로와 고급 비주얼 효과를 구현할 수 있습니다. VFX Graph는 아티스트가 HDRP의 렌더링 기능을 활용하여 실시간 비주얼 효과를 만들 수 있도록 HDRP를 보완합니다.

커스텀 HLSL 블록과 오퍼레이터

Unity 6에는 VFX Graph에서 바로 [커스텀 HLSL](#) 코드를 작성할 수 있는 새로운 [블록](#)과 [오퍼레이터](#)가 추가되었습니다. 이를 통해 복잡한 동작을 하나의 노드에서 스크립팅할 수 있습니다. 노드 안에 커스텀 HLSL 코드를 임베드하거나 외부 파일에서 로드하여 코드를 재사용할 수 있습니다.

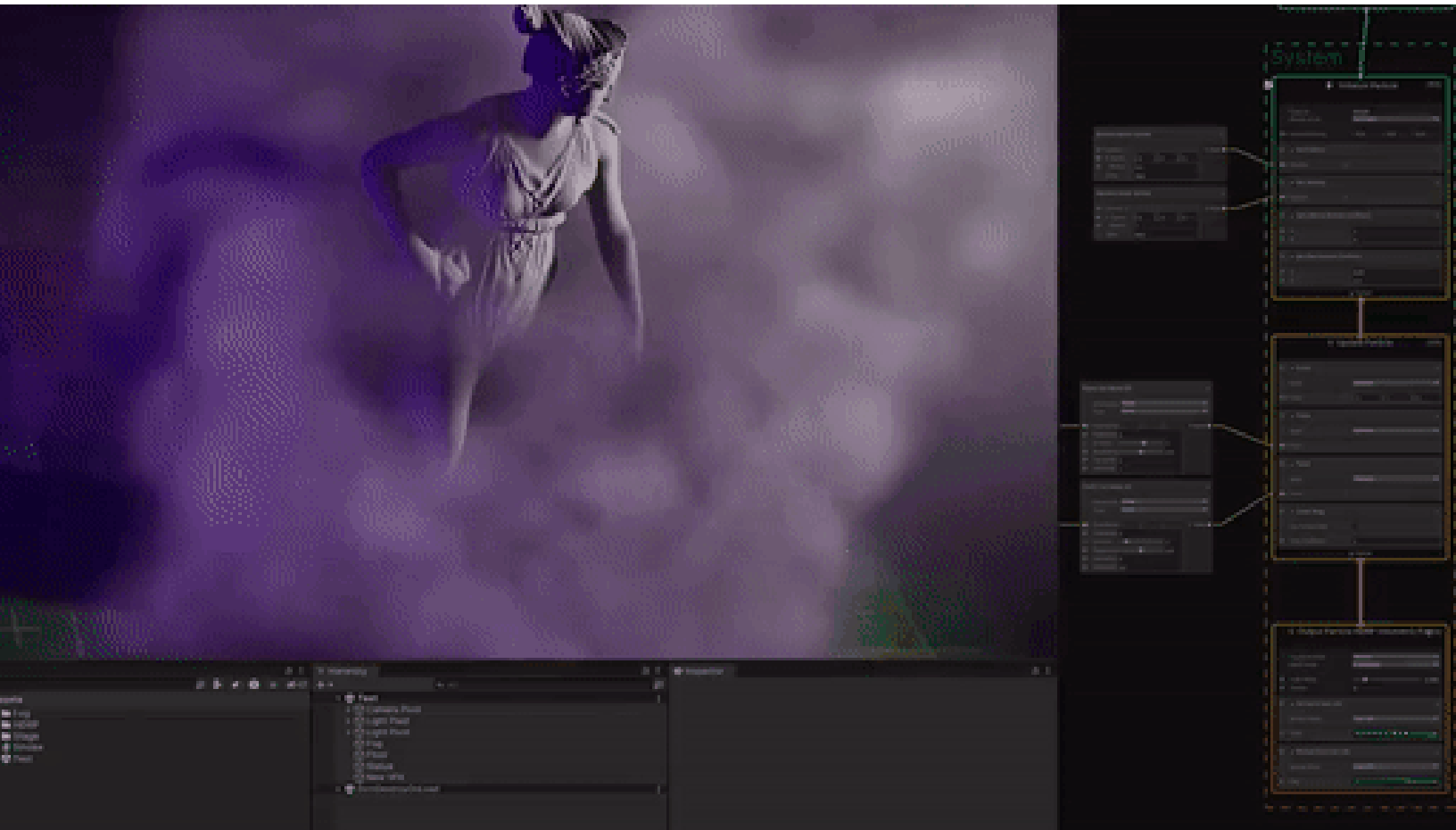
이 기능을 사용하여 커스텀 셰이더를 작성하거나 Unity의 빌트인 기능을 확장하여 특별한 효과를 구현하세요. 예를 들어, 이웃 탐색 알고리즘을 사용하여 커스텀 군집 동작을 구현하거나, 버퍼 읽기를 사용하여 비주얼 효과를 오디오 트리거와 동기화할 수 있습니다.



HLSL 블록을 사용하여 만든 효과.

볼류메트릭 안개 출력

Output Particle HDRP Volumetric Fog 컨텍스트는 HDRP의 [Local Volumetric Fog](#)를 샘플링하여 VFX Graph가 볼류메트릭 광원에 직접 안개를 삽입할 수 있도록 합니다. 이를 통해 VFX Graph 파티클과 씰 조명 간의 상호 작용이 향상되어 사실감이 부여됩니다. 다이내믹 안개 효과, 사실적인 구름, 연기나 불을 생성하는 데 사용하세요.



Output Particle HDRP Volumetric Fog 출력 노드

레이트레이싱 지원

Unity 6의 VFX Graph 파티클은 이제 HDRP에서 다음과 같은 레이트레이싱 기능을 지원합니다.

- 레이트레이싱 반사
- 굴절, 그림자
- 앰비언트 오클루전
- 전역 조명

사각형, 삼각형, 팔각형을 사용하는 VFX Graph에서 레이트레이싱을 활성화할 수 있습니다. HDRP 씬에서 레이트레이싱을 사용하려면 [레이트레이싱 시작하기](#)를 참조하세요. 참고: VFX Graph는 메시 또는 스트립을 사용한 레이트레이싱을 지원하지 않습니다.

HDRP 프로젝트에 VFX Graph를 통합하는 방법을 알아보려면 [집중 탐구 가이드: Unity에서 고급 시각 효과 만들기\(Unity 6 에디션\)](#) 전자책을 참조하세요.

최적화

복잡한 게임 환경은 높은 성능을 요구하기 때문에 렌더 파이프라인을 최적화하는 것이 애플리케이션에 매우 중요합니다. 게임이 모듈식 요소나 프리팹을 사용한다면 각종 기법을 사용하여 CPU 부하를 줄이고 렌더링 효율을 높일 수 있습니다.

스크립터블 렌더 파이프라인 배치

SRP 배치는 드로우 콜을 준비하고 디스패치하는 데 필요한 CPU 오버헤드를 줄여서 [드로우 콜을 최적화](#)합니다. SRP 배치는 드로우 콜의 횟수를 줄이는 대신 드로우 콜 사이의 렌더 상태 변화를 최소화합니다.

SRP 배치는 bind 및 draw GPU 커맨드를 **SRP 배치**로 그룹화하여 동일한 셰이더 배리언트를 공유하는 머티리얼의 성능을 향상합니다.

최적의 성능을 위해서는 각 SRP 배치에 가급적 많은 bind 및 draw 커맨드를 포함해야 합니다. 셰이더 배리언트가 적을수록 배치 효율이 극대화되고 여러 머티리얼이 동일한 셰이더를 사용할 수 있습니다.

SRP 배치가 작동하는 방식에 대한 자세한 내용은 이 [기술 자료 페이지](#)를 참조하세요.

BatchRendererGroup

BRG(BatchRendererGroup)는 SRP(스크립터블 렌더 파이프라인) 및 SRP 배치를 사용하는 프로젝트에서 많은 수의 오브젝트를 효율적으로 처리하도록 설계된 Unity의 렌더링 API입니다.

BRG는 기존 게임 오브젝트 기반 렌더링을 사용하는 대신 렌더링 데이터를 직접 제어할 수 있도록 합니다. 따라서 기존 게임 오브젝트의 비용 부담이 높은 DOTs 엔티티, 절차적 환경 및 커스텀 터레인에 적합합니다.

BRG는 여러 오브젝트를 하나의 드로우 콜로 배치하여 CPU 오버헤드를 줄이고 성능을 최적화합니다. GPU에 지속적인 버퍼를 유지하여, 빈번한 CPU 개입 없이 오브젝트 데이터를 저장·및 업데이트합니다.

BRG를 사용하려면 **SRP 배치** 호환성을 활성화하고, **Project Settings > Graphics**에서 BRG 셰이더 배리언트를 그대로 두고, **Player Settings**에서 Allow 'unsafe' Code 옵션을 통해 안전하지 않은 코드를 허용해야 합니다.

BRG는 대규모 인스턴싱이 요구되는 씬(밀집한 초목, 화면에서 여러 오브젝트가 렌더링되는 경우, 다이내믹 LOD 처리)에서 특히 효과적입니다. BRG는 새로운 GPU 상주 드로어와 통합됩니다.

GPU 상주 드로어

Unity 6의 GPU 상주 드로어는 게임 오브젝트의 렌더링 효율을 크게 향상합니다. 이 새로운 기능은 [BatchRendererGroup API](#)를 사용하여 인스턴싱을 더 효율적으로 처리하는 빠른 경로를 통해 MeshRenderer를 배치 및 렌더링합니다. GPU 상주 드로어는 오브젝트별 처리 비용을 줄여 드로우 콜이 많은 씬에서 CPU 병목 현상을 완화합니다.

이 기능은 스크립터블 렌더 파이프라인(URP 및 HDRP)과 호환되며 컴퓨터 셰이더를 지원하는 그래픽스 API가 필요합니다(OpenGL/GLES에서는 사용할 수 없음).

씬이 복잡할수록 GPU 상주 드로어의 성능 이점이 커집니다. 씬에 디테일이 많고 인스턴싱된 오브젝트가 많을수록 더 큰 성능 향상을 얻을 수 있습니다.

GPU 상주 드로어를 활성화하는 방법은 다음과 같습니다.

- **Project Settings > Graphics**로 이동하여 **BatchRendererGroup Variants**를 **Keep All**로 설정합니다.
- **HDRP Render Pipeline Asset**을 찾습니다. **GPU Resident Drawer Mode**를 찾아서 **Instanced Drawing**으로 설정합니다.
- 게임 오브젝트의 머티리얼에서 **Enable GPU Instancing**이 선택되어 있는지 확인합니다.

GPU 상주 드로어를 최적화하려면 **Edit > Project Settings > Player**에서 프로젝트에 대해 또는 인스펙터에서 인스턴스별로 **static batching**을 비활성화합니다. static batching이 활성화되어 있으면 GPU 상주 드로어가 작동하지 않습니다.

GPU 상주 드로어를 사용할 때는 추가 셰이더 배리언트를 컴파일하느라 빌드 시간이 증가할 수 있습니다. 이 기능은 표준 MeshRenderer만 지원하며 스킨드 메시 렌더러, VFX Graph, 파티클 시스템, 기타 효과 렌더러는 처리하지 않습니다.

GPU 상주 드로어를 활성화하면 여러 게임 오브젝트가 동일한 메시를 공유하는 씬에서 드로우 콜을 크게 줄일 수 있습니다. 10,000번 인스턴싱된 프리팹을 프로파일러에서 비교해 보세요.

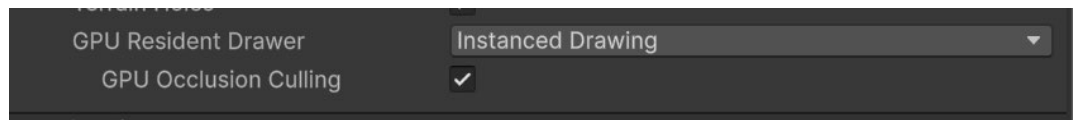


GPU 상주 드로어를 사용하는 경우와 사용하지 않는 경우의 수치 비교.

GPU 오클루전 컬링

씬에서 GPU 상주 드로어를 사용한다면 여기에 더해 **GPU 오클루전 컬링**을 사용하여 화면에 실제로 보이지 않는 GPU 인스턴스를 제거할 수 있습니다. 이를 통해 오버드로우가 많이 발생하는 씬에서 하나의 설정으로 GPU 성능을 개선할 수 있습니다.

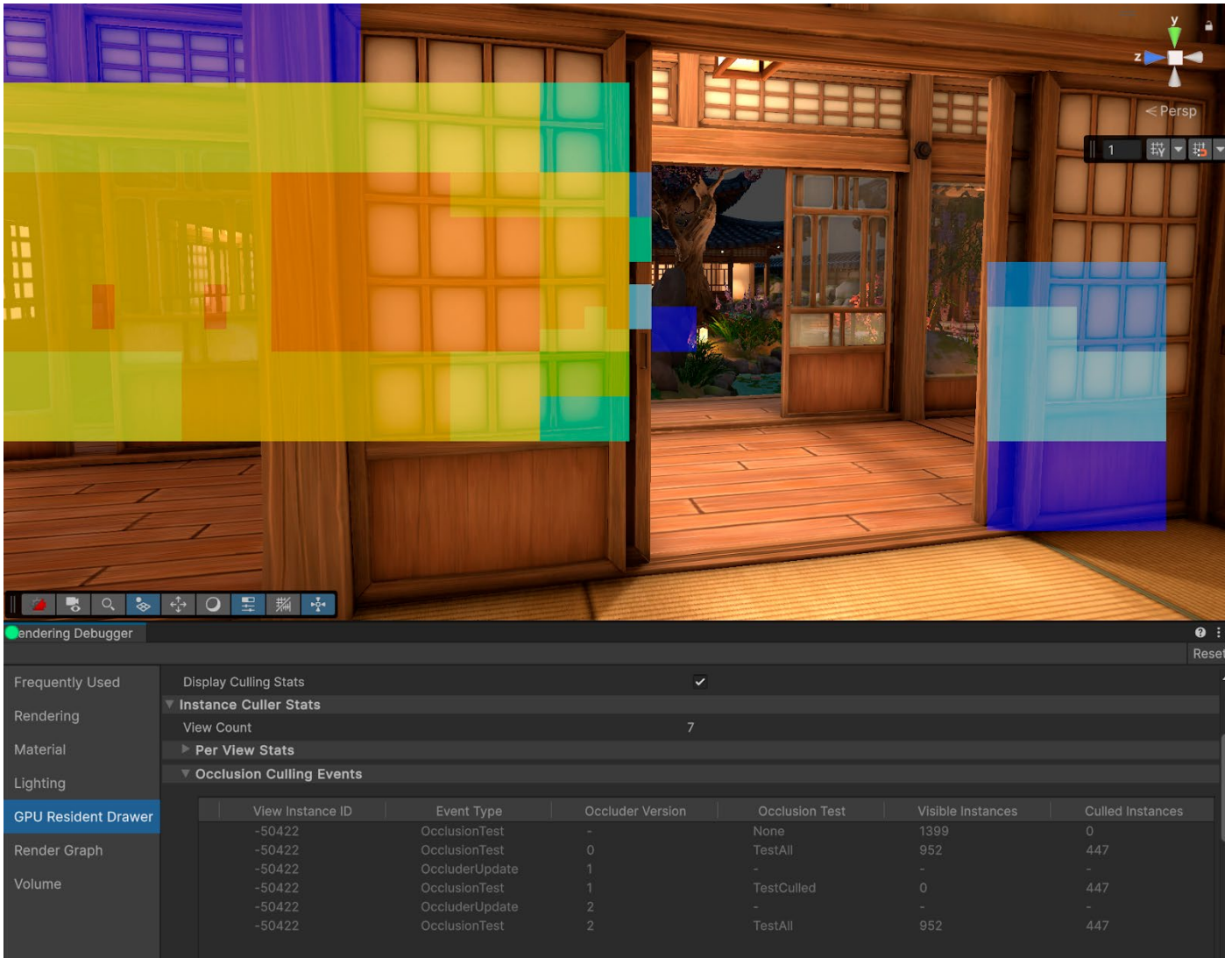
GPU 오클루전 컬링을 활성화하려면 먼저 GPU 상주 드로어 모드를 활성화한 후에 HDRP 에셋으로 이동합니다. **Rendering** 섹션으로 이동한 다음 **GPU Occlusion Culling**을 활성화합니다.



GPU Occlusion Culling을 활성화합니다.

작업 중인 씬에 GPU 오클루전 컬링이 효과적인지 확인하려면 **Window > Analysis > Rendering Debugger**로 이동하여 **GPU Resident Drawer > Occlusion Test Overlay**를 선택하세요.

RenderDebugger는 화면을 히트맵으로 표시하여 씬의 어느 부분이 가장 큰 성능 이점을 얻는지 보여 줍니다. **파란색** 영역은 컬링된 인스턴스가 적음을 나타내고, **빨간색** 영역은 컬링된 인스턴스가 많음을 의미하며, 그 사이에는 색상 그라데이션이 표시됩니다.



GPU 상주 드로잉은 URP에서도 작동합니다.

자세한 내용은 [Unity 고급 사용자를 위한 유니버설 렌더 파이프라인 소개](#) 가이드북을 참조하세요.

DX12에서 에디터를 통한 그래픽스 잡 지원

이제 DX12를 사용하는 경우 그래픽스 잡이 Unity 에디터에서도 지원됩니다. 이를 통해 복잡한 환경에서 에디터로 작업할 때 씬 뷰와 플레이 모드 렌더링 성능이 향상됩니다.

이 기능을 활성화하려면 **Graphics API**로 **DirectX 12**를 설정합니다. 그런 다음 **Player** 설정에서 **Graphics Jobs**를 선택하고 **Preferences/Settings > Jobs**의 **Jobs** 설정에서 **Allow Graphics Jobs in Editor**를 활성화합니다.

DX12(DirectX 12) 같은 최신 그래픽스 API는 커맨드 버퍼를 멀티스레드로 기록 및 제출하여 CPU 사용률을 개선합니다. Unity의 그래픽스 잡 시스템은 드로우 콜을 처리할 때 CPU 병목 현상을 줄여서 이 프로세스를 최적화합니다.

메인 스레드는 드로우 커맨드를 대기열에 추가하고, 렌더링 스레드는 워커 스레드를 생성하여 이를 GPU 커맨드 버퍼로 변환 및 제출합니다. 이 업데이트를 통해 고성능 렌더링 및 대규모 씬에서 CPU-GPU 워크로드가 효율적으로 배분됩니다.

DX12 그래픽스 잡은 DirectX 12를 사용하는 Windows 플랫폼의 에디터에서 지원됩니다. 런타임 빌드에서는 Windows(DX12) 및 콘솔 플랫폼에서 지원됩니다.

Variable Rate Shading(Unity 6.1)

VRS(Variable Rate Shading)는 화면의 각 부분별로 셰이딩의 디테일을 조절하는 기능입니다. 픽셀 또는 오브젝트 수준에서 셰이딩 품질을 제어하면 비주얼 품질에 큰 영향을 주지 않으면서 GPU 성능을 향상할 수 있습니다.

예를 들어, 레이싱 게임에서는 플레이어의 차량과 세밀한 주변 환경에는 고품질 셰이딩을 사용하고, 모션 블러가 적용된 도로처럼 눈에 덜 띄는 영역에는 저품질 셰이딩을 적용하여 리소스를 효율적으로 사용할 수 있습니다.

Unity 6.1에서는 [Variable Rate Shading](#)을 위한 새로운 API가 도입되었으며, Windows(DirectX 12), Vulkan 및 호환 콘솔이 지원됩니다.

Variable Rate Shading이 작동하는 방식을 알아보려면 [Shading Rate 샘플 프로젝트](#)를 살펴보세요. 자세한 내용은 [이 Discussions 페이지](#)를 참조하세요.



Variable Rate Shading은 화면의 각 부분별로 셰이딩 품질을 조정합니다.

다음 단계

이 가이드가 여러분의 프로젝트에서 HDRP를 활용하는 데 도움이 되길 바랍니다. 더 자세히 살펴보고 싶다면 Unity Hub에서 **3D 샘플 프로젝트**를 다운로드하세요. 이 가이드에 대한 질문이나 피드백이 있다면 [포럼 스레드](#)에 알려 주시기 바랍니다.

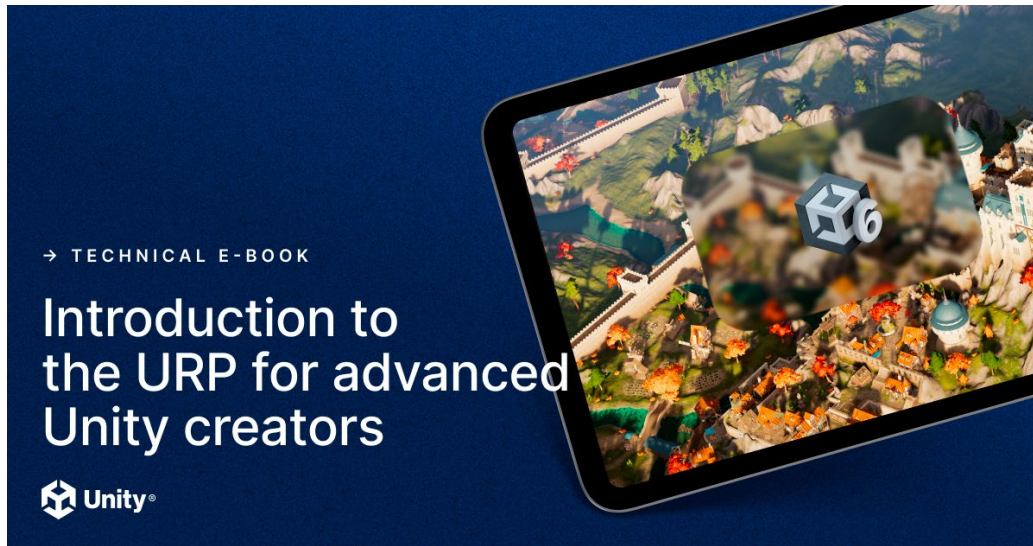
아래에 소개된 추가 자료를 꼭 확인해 보세요. [Unity 블로그](#) 나 [HDRP 커뮤니티 포럼](#)에서도 언제든지 팁을 얻을 수 있습니다.

유니티는 아티스트와 개발자에게 실시간 콘텐츠를 제작할 수 있는 최고의 툴을 제공하기 위해 노력하고 있습니다. 게임 월드와 환경을 제작하는 일은 예술인 동시에 과학입니다. 미적 감각과 과학의 조화로 환상적인 결과물을 얻어 보세요.

더 많은 자료

- [‘HDRP로 정확도 높은 게임 그래픽스 구현하기’](#) 영상에서도 HDRP 기능을 소개하고 있습니다.
- 빌트인 렌더 파이프라인에서 마이그레이션하는 경우, [이 표](#)에서 비교한 두 렌더 파이프라인 간의 기능 차이를 살펴보면 좋습니다.
- [HDRP 기술 자료](#)에서 파이프라인의 모든 기능에 대한 자세한 내용을 확인할 수 있습니다.
- [이 블로그 게시물](#)에서 성능 개선을 위한 HDRP 설정을 자세히 알아보세요.

- 새로운 물 시스템에 대해 자세히 알아보려면 이 [블로그 게시물](#) 또는 물 시스템 [기술 자료](#)를 살펴보세요. 또한 이 [소개 영상](#)에서 물 시스템 기능의 개요를 알아볼 수도 있습니다.
- [Unity 개발](#)과 [테크니컬 아트](#)에 대해 더 알고 싶으신가요? [Unity 베스트 프랙티스 허브](#)를 방문하여 실행 가능한 팁과 베스트 프랙티스를 다양하게 제공하는 문서와 전자책을 살펴보고, 더 빠르게 더 많은 것을 구현해 보세요.



프로그래머, 아티스트, 테크니컬 아티스트, 디자이너를 위한 유니티의 기술 전자책에서 더 자세한 내용을 살펴보세요.



unity.com/kr