



게임엔진프로그래밍응용

9. 탑다운 슈터 게임 1

청강문화산업대학교 게임콘텐츠스쿨

반 경 진

공지사항

공지사항

- 2023학년도 1학기 방역 및 학사운영 방안
<https://www.ck.ac.kr/archives/193175>
- 2023학년도 1학기 국가공휴일 및 대학 행사 수업 대체 일정 공지
<https://www.ck.ac.kr/archives/193109>
- 다음주 5.10(수) 개교기념일 휴강

2023학년도 1학기 중간 수업평가

2023학년도 1학기 중간 수업평가

- 평가 일정

2023.04.24.(월) ~ 2023.05.07.(일)

- 평가 방법

<https://www.ck.ac.kr/archives/194753>

온라인 수업 저작권 유의 사항

온라인 수업 저작권 유의 사항

온라인수업 저작권 유의사항 안내



**강의 저작물을 다운로드, 캡처하여
교외로 유출하는 행위는
불 법 입 니 다**

저작권자의 허락 없이 저작물을 복제, 공중송신 또는 배포하는 것은
저작권 침해에 해당하며 저작권법에 처벌받을 수 있습니다.

강의 동영상과 자료 일체는 교수 및 학교의 저작물로서 저작권이 보호됩니다.
수업자료를 무단 복제 또는 배포, 전송 시 민형사상 책임을 질 수 있습니다.

Index

1

게임 컨셉 및 기능

2

레벨과 라이트맵

3

플레이어 캐릭터와 애니메이션

4

플레이어 추적 카메라

게임 컨셉 및 기능

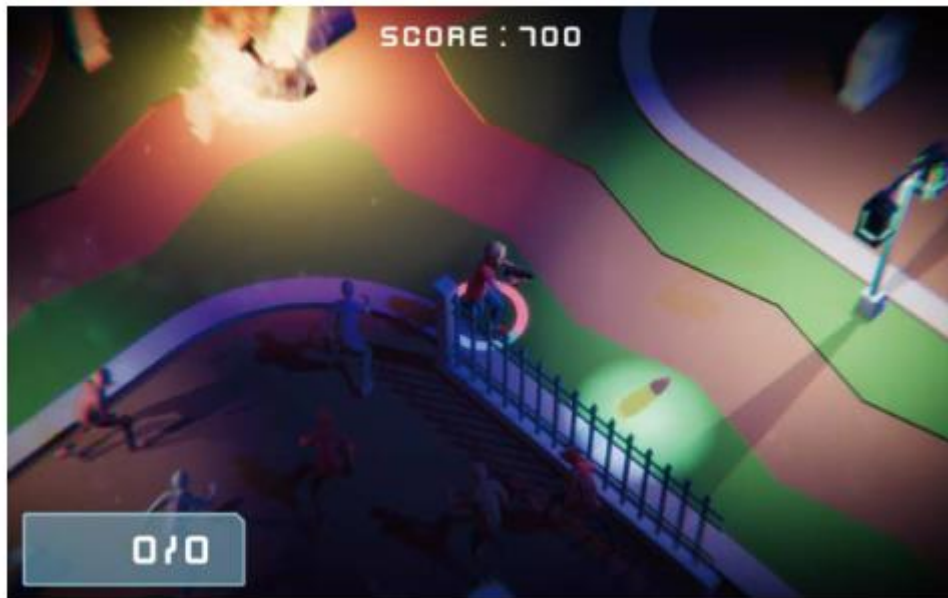
게임 컨셉 및 기능

- **좀비 서바이버**

웨이브형 좀비 탑다운 슈터 게임

1. 좀비는 일정 주기로 생성, 시간이 지날수록 한 번에 생성되는 좀비 수 증가
2. 좀비는 플레이어의 위치를 주기적으로 파악하고 언제나 최적의 경로를 찾아 플레이어를 추적
3. 좀비의 이동 속도와 공격력, 체력은 랜덤으로 지정
4. 강하고 빠른 좀비일수록 피부 색깔이 붉어짐
5. 플레이어의 체력은 캐릭터를 따라다니는 원형바로 표시
6. 탄알 아이템과 체력아이템(주기적으로 플레이어 근처의 랜덤 위치에서 생성 후 일정시간 뒤에 사라짐)
7. Post-Processing 효과를 사용하여 게임 화면 보정
8. 키보드를 사용하여 캐릭터 조작
 - 캐릭터 회전(A, D 또는 Left, Right)
 - 캐릭터 전진/후진(W, S 또는 Up, Down)
 - 탄알 발사(마우스 왼쪽 버튼)
 - 재장전(R)

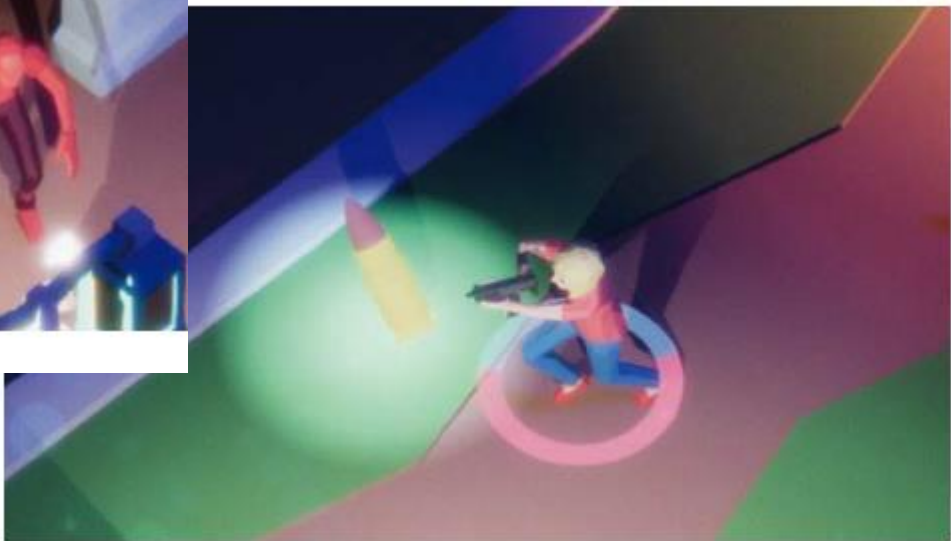
게임 컨셉 및 기능



▶ 플레이어를 추적하는 인공지능 좀비



▶ 강한 좀비일수록 피부가 붉다



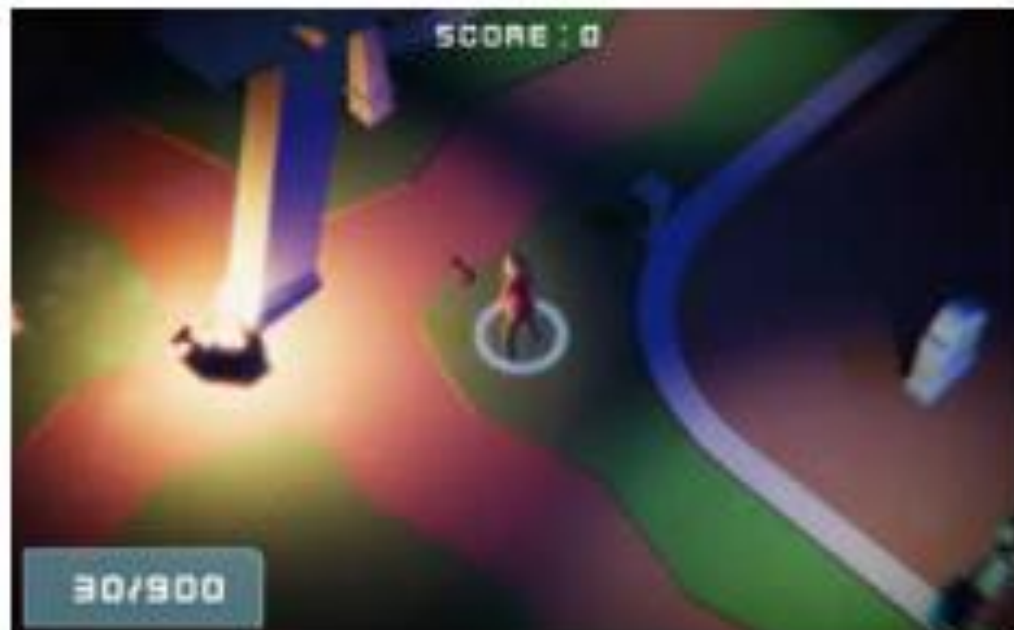
▶ 랜덤한 주기와 위치로 배치되는 아이템

게임 컨셉 및 기능

후처리 X



후처리 O



▶ 후처리에 의한 영상미 차이

레벨과 라이트맵

레벨과 라이트맵

- 좀비 서바이버

1. [Zombie.zip](#) 압축해제
2. 프로젝트 열기
3. 씬 생성
4. 씬에서 Directional Light 삭제
5. Level Art Prefab 배치

레벨과 라이트맵

- 레벨(Level)

1. 맵으로서의 레벨이란 D&D등 초창기 RPG에서 던전의 단계(Level)를 게임 난이도와 테마로 구분했던 것에서 유래
2. 레벨 디자인 = 맵 디자인 (레벨 디자이너 = 매퍼)
3. 레벨 콘셉트 디자인 (맵의 전체적인 주제를 설정)
4. 구조, 공간 디자인 (콘셉트를 구체화된 공간, 구조로 만드는 디자인)
5. 환경, 배경 디자인 (텍스처, 오브젝트, 조명 등 공간을 멋지게 꾸미는 디자인으로, 쉽게 말해 인테리어 디자인)
6. 레벨 스크립트 디자인 (특정 지점에서 몬스터가 리젠되거나, 함정이 발동되는 등 맵의 특수한 기믹을 설치하는 디자인)

레벨과 라이트맵

- 조명(Lighting)

- 직접 조명(Direct Lighting)

- 방출되어 표면에 한 번 닿은 후 센서(예: 눈의 망막 또는 카메라)에 직접 반사되는 광원

- 간접 조명(Indirect Lighting)

- 표면에 여러 번 닿는 광원, 스카이 라이트 등을 포함하여 궁극적으로 센서에 반사되는 다른 모든 광원

- 실시간 조명(Real-time Lighting)

- 런타임 시점에 계산하는 조명

- 베이킹된 조명(Baked Lighting)

- 조명 계산을 미리 수행하고 결과를 조명 데이터로 저장한 후 런타임 시점에 적용되는 조명

- 전역 조명(Global illumination)

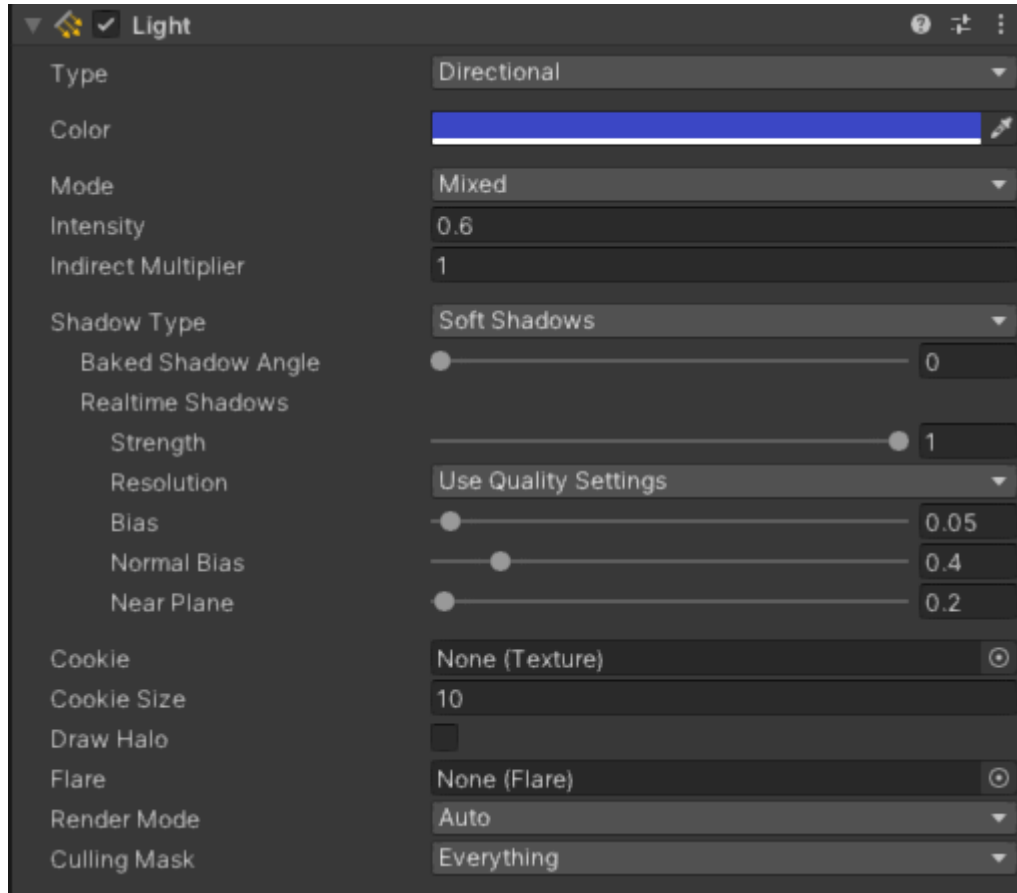
- 전역 조명은 직접 조명과 간접 조명을 모두 모델링하여 사실적인 조명 결과를 제공하는 기술 그룹

레벨과 라이트맵

- 광원(Light sources)
 - Light 컴포넌트
 - 발광 머터리얼(Emissive materials)
 - 주변광(Ambient lighting)

레벨과 라이트맵

- Light 컴포넌트



레벨과 라이트맵

- Type of Light

1. Spot

스팟 광원, 썬의 한 점에 위치하여 원뿔 모양으로 빛을 발산하는 광원

2. Directional

방향 광원, 무한히 멀리 위치하여 한 방향으로만 빛을 발산하는 광원

3. Point

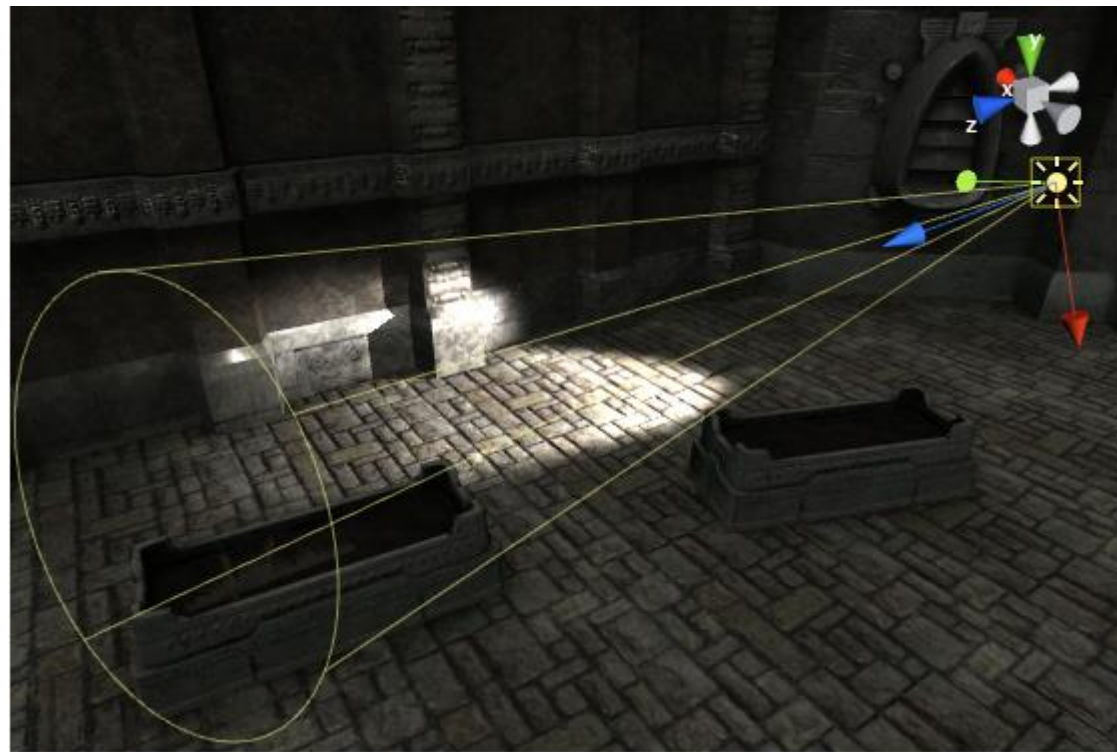
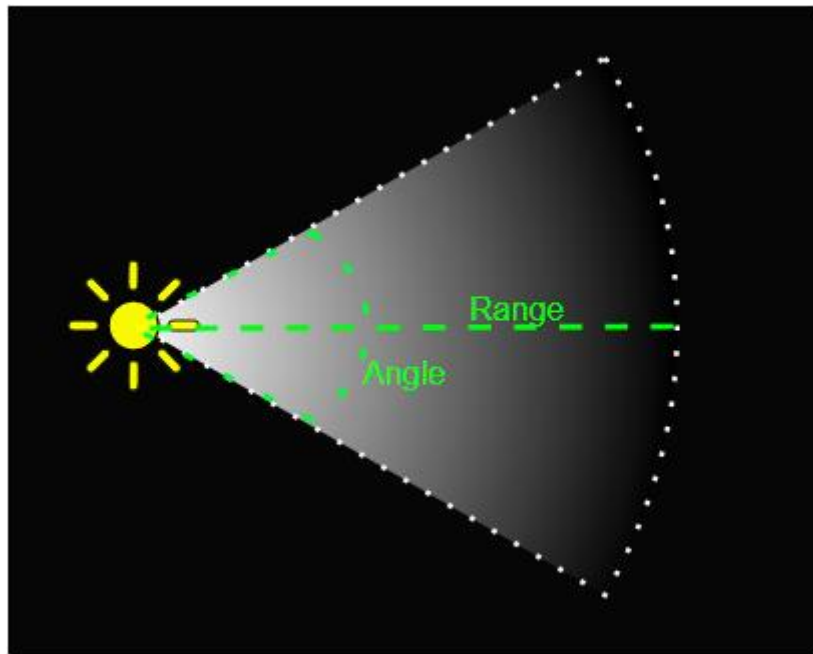
점 광원, 썬의 한 점에 위치하여 모든 방향으로 균등하게 빛을 발산하는 광원

4. Area

영역 조명, 썬에서 사각형으로 정의하며 표면 영역 전체에 걸쳐 균등하게 모든 방향으로 빛을 방출하지만 사각형의 한쪽 면에서만 빛을 방출하는 광원

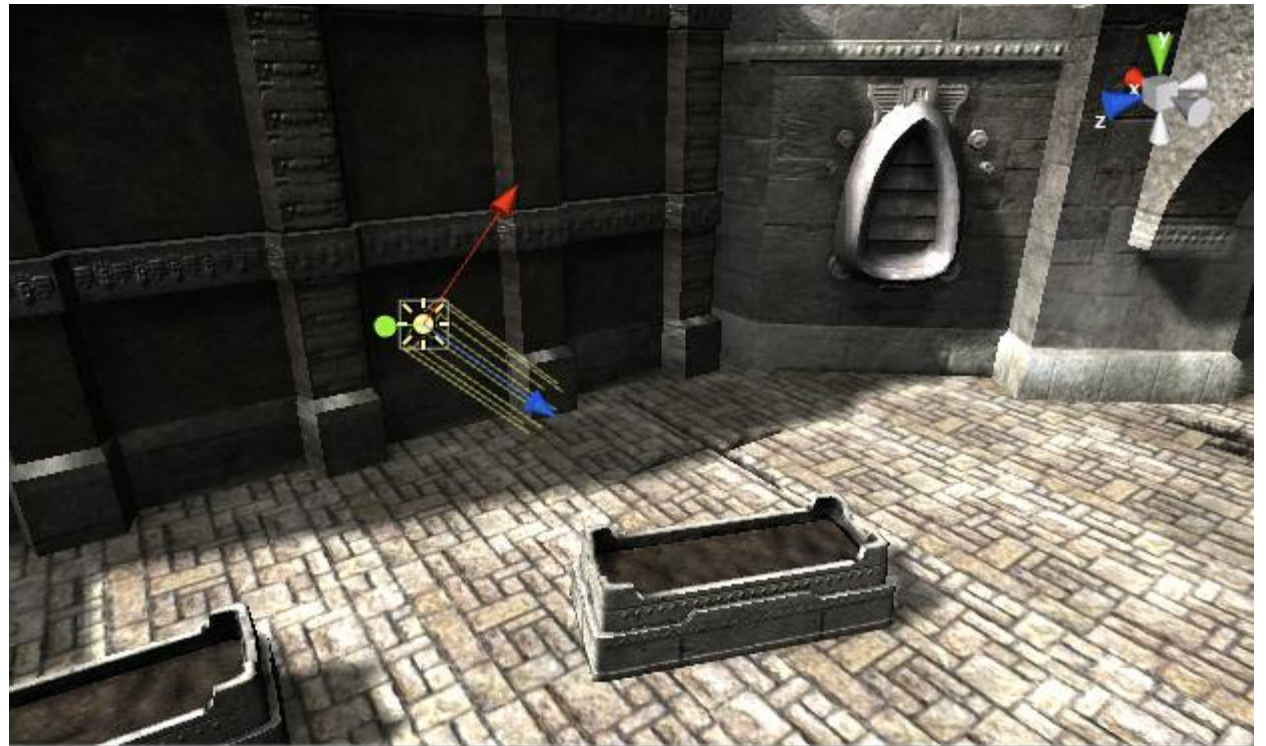
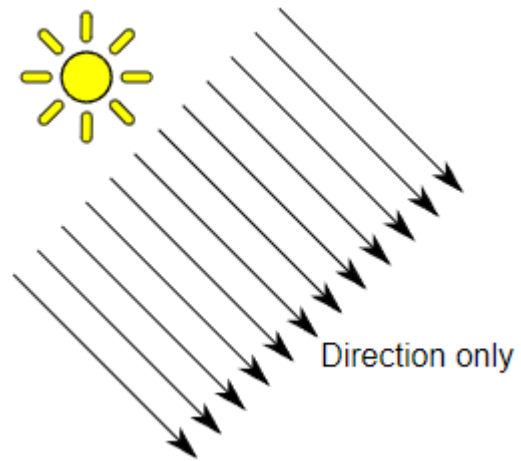
레벨과 라이트맵

- Spot Lights



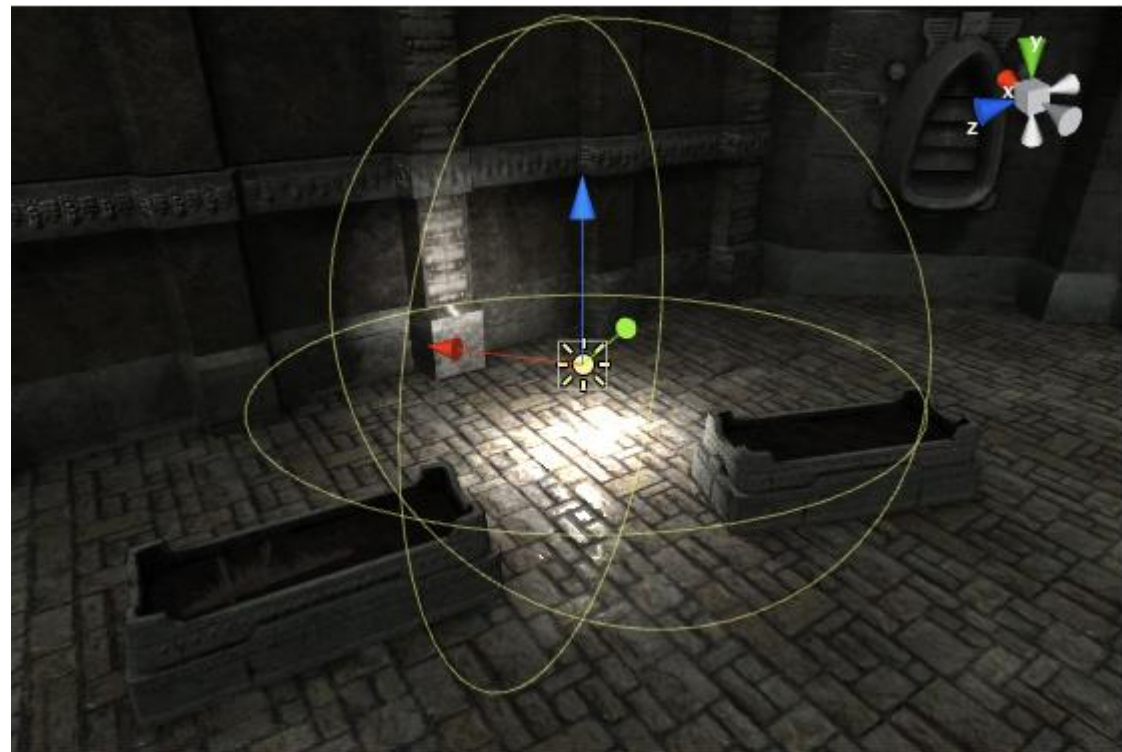
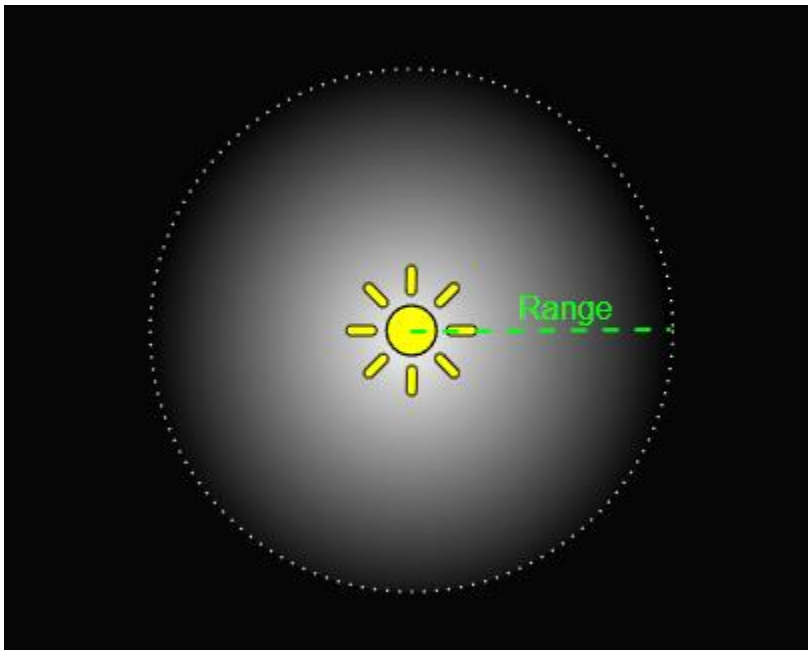
레벨과 라이트맵

- Directional Lights



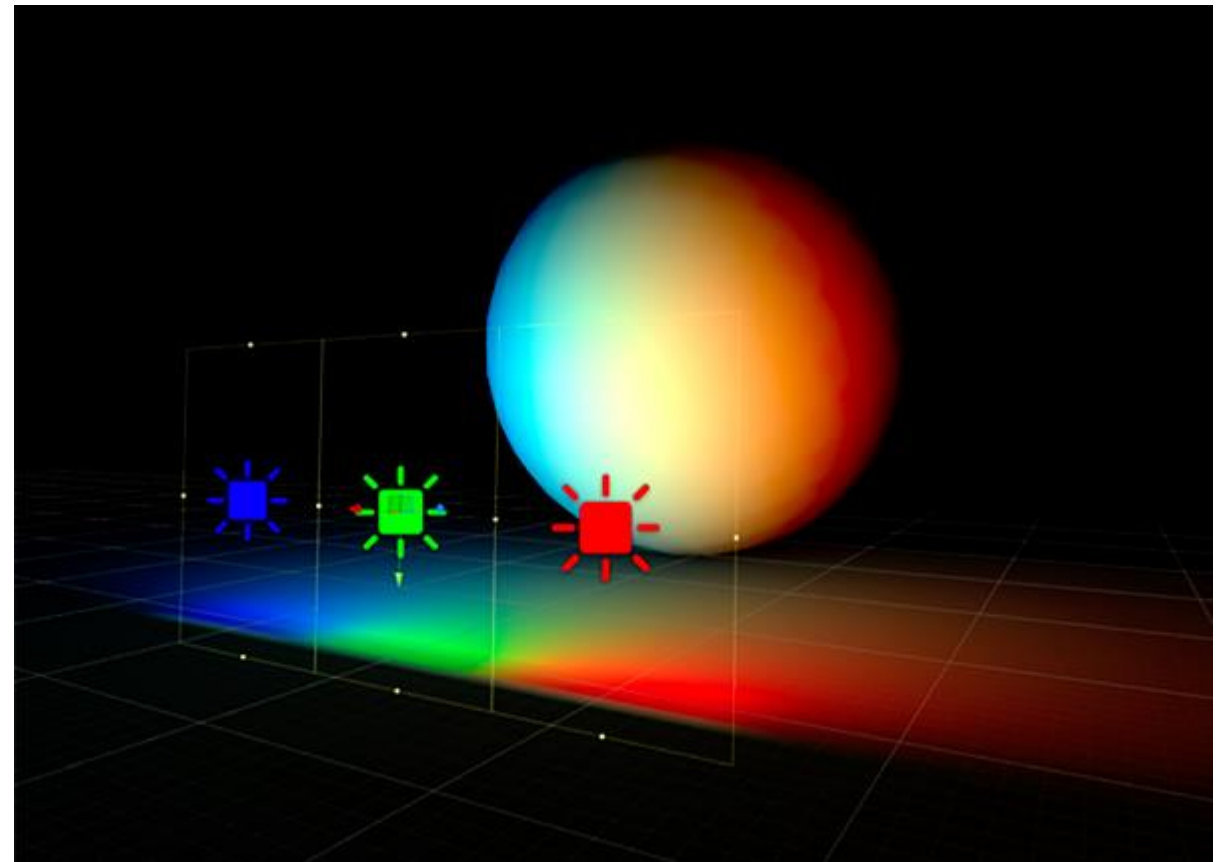
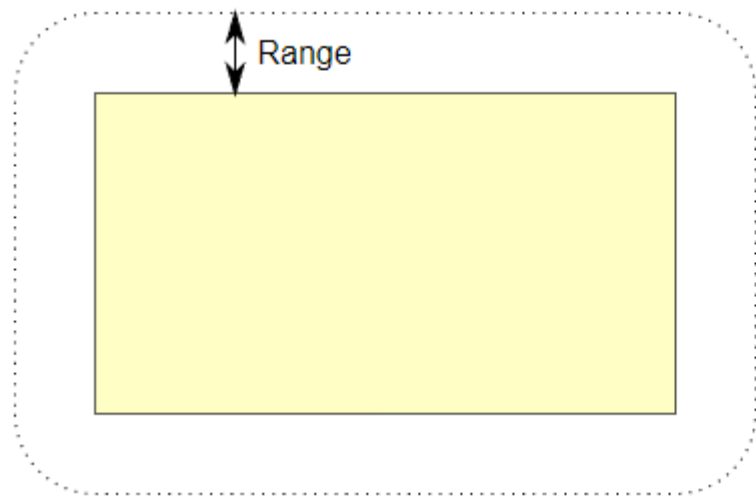
레벨과 라이트맵

- Point Lights



레벨과 라이트맵

- Area Lights



레벨과 라이트맵

- Mode of Lights
 - Baked
 - Realtime
 - Mixed

레벨과 라이트맵

- 그림자
 - No Shadows
 - Hard Shadows
 - Soft Shadows

레벨과 라이트맵

- 쿠키



레벨과 라이트맵

- 추가 설정
 - Draw Halo
 - Flare
 - Render Mode
 - Culling Mask

레벨과 라이트맵

- 발광 머터리얼(Emissive materials)



레벨과 라이트맵

- 주변광(Ambient lighting) 또는 디퓨즈 환경광

씬 전체에 존재, 특정 소스 오브젝트에서 나오지 않는 광원.

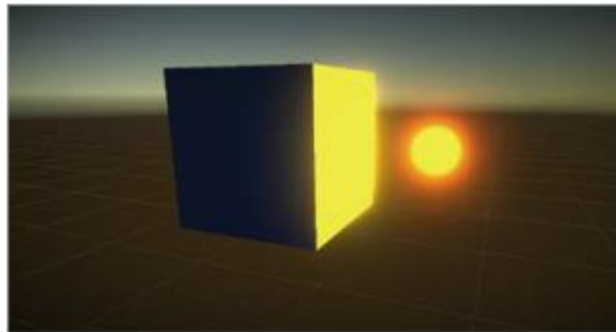
씬의 전체적인 외양과 밝기에 영향.

개별 광원을 조정하지 않고 씬의 전체적인 밝기를 높여야 하는 경우

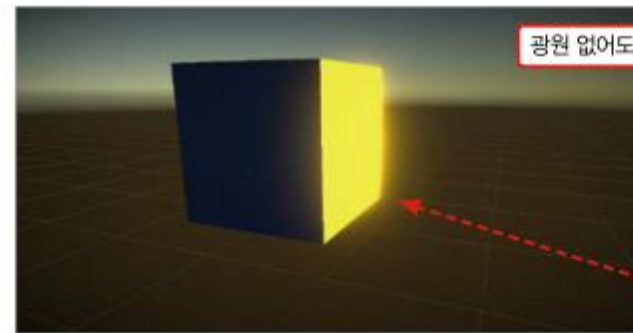
주변광 설정은 라이팅 창에서

레벨과 라이트맵

- 라이트맵

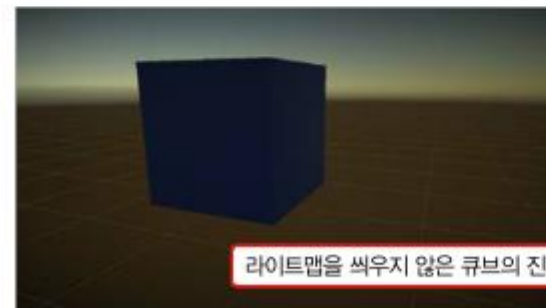


▶ 노란 빛을 받은 큐브



광원 없어도 큐브 벽면이 빛남

큐브에 라이트맵을 덧씌움



라이트맵을 씌우지 않은 큐브의 진짜 모습



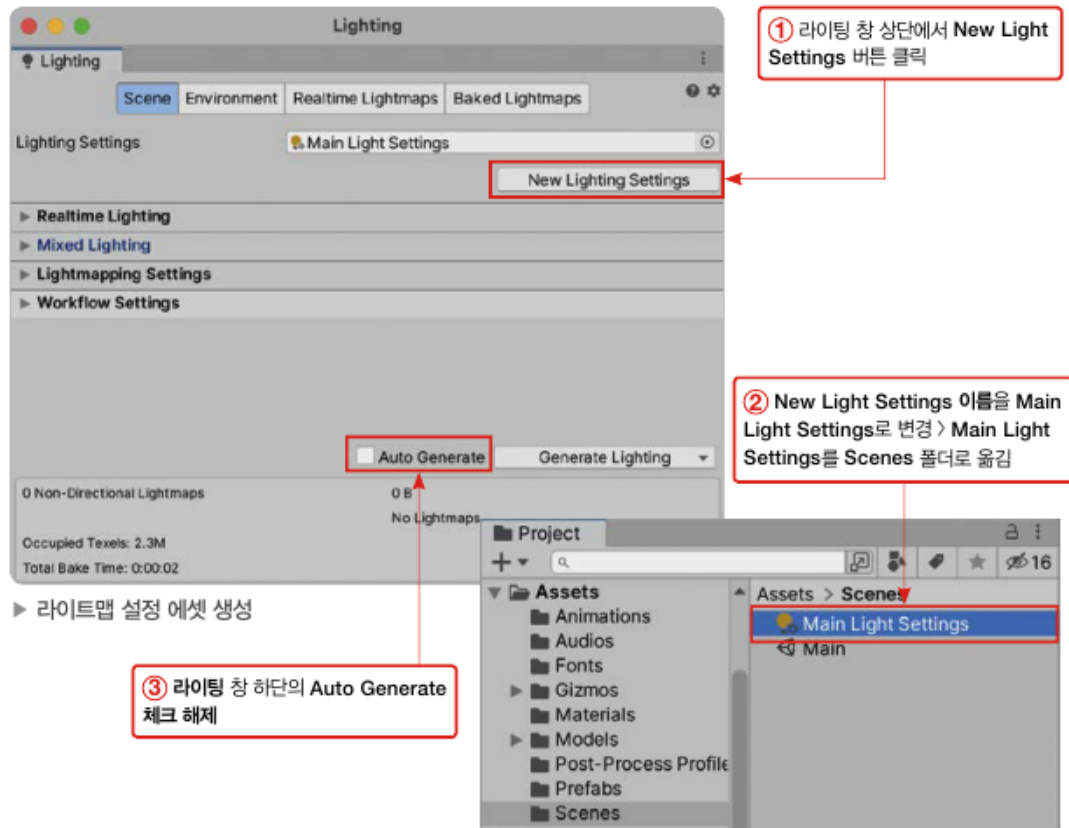
큐브의 라이트맵

▶ 라이트맵이 입혀진 큐브

레벨과 라이트맵

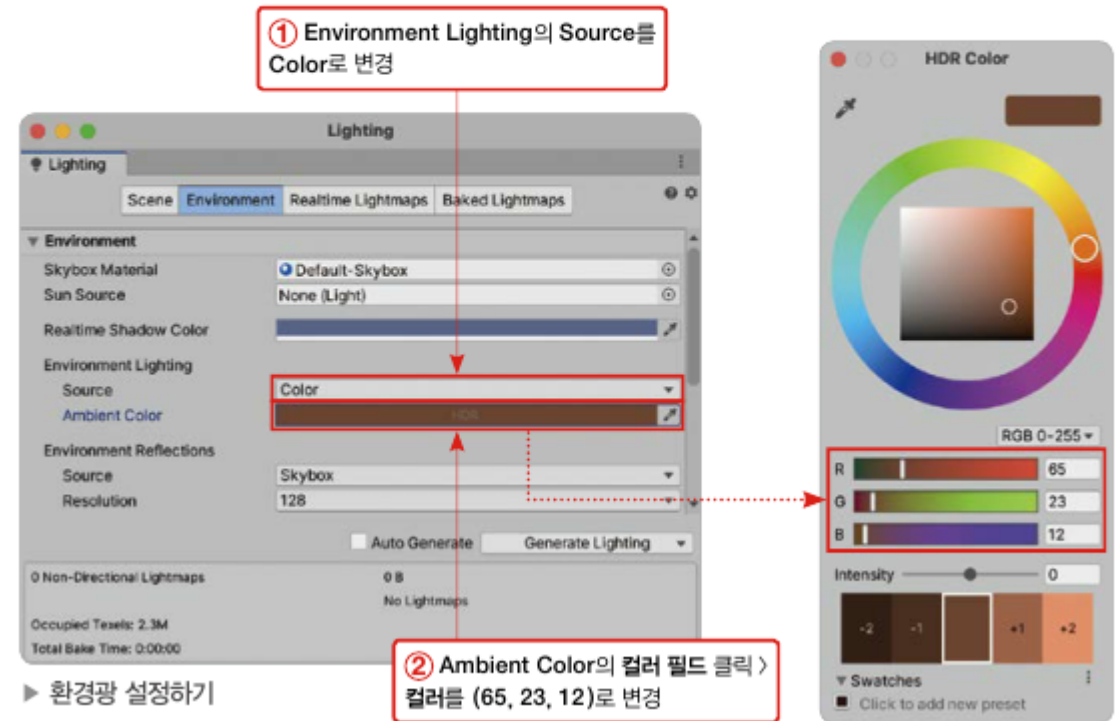
[과정 01] 라이트맵 설정 에셋 생성

- ① 라이팅 창 상단에서 **New Light Settings** 버튼 클릭 > 라이트 설정 에셋 **New Light Settings**가 생성됨
- ② **New Light Settings** 이름을 **Main Light Settings**로 변경 > **Main Light Settings**를 **Scenes** 폴더로 옮김
- ③ 라이팅 창 하단의 **Auto Generate** 체크 해제



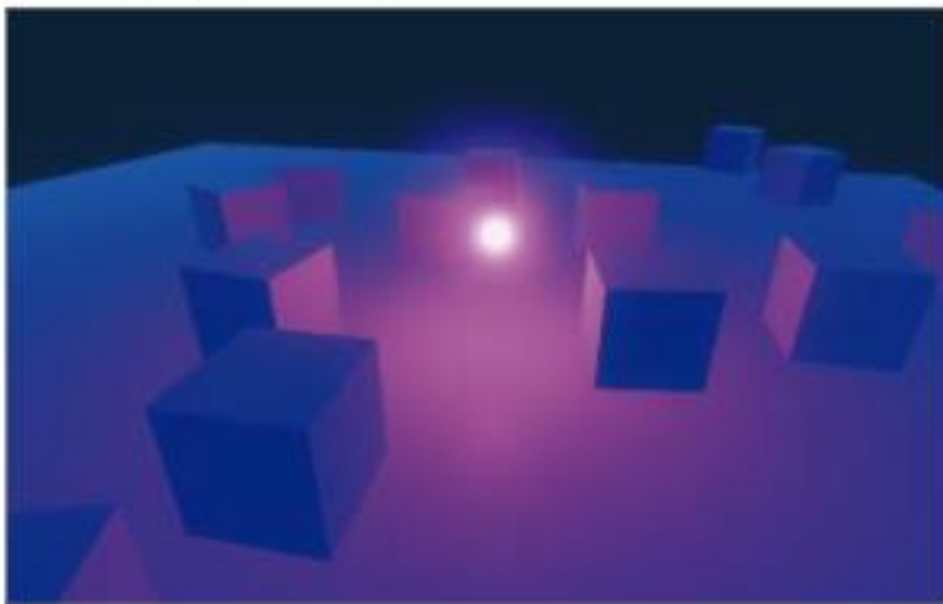
[과정 02] 환경광 설정하기

- ① 라이팅 창 상단의 **Environment** 탭 클릭 > **Environment** 패널에서 **Source**를 **Color**로 변경
- ② **Ambient Color**의 컬러 필드 클릭 > 컬러를 (65, 23, 12)로 변경

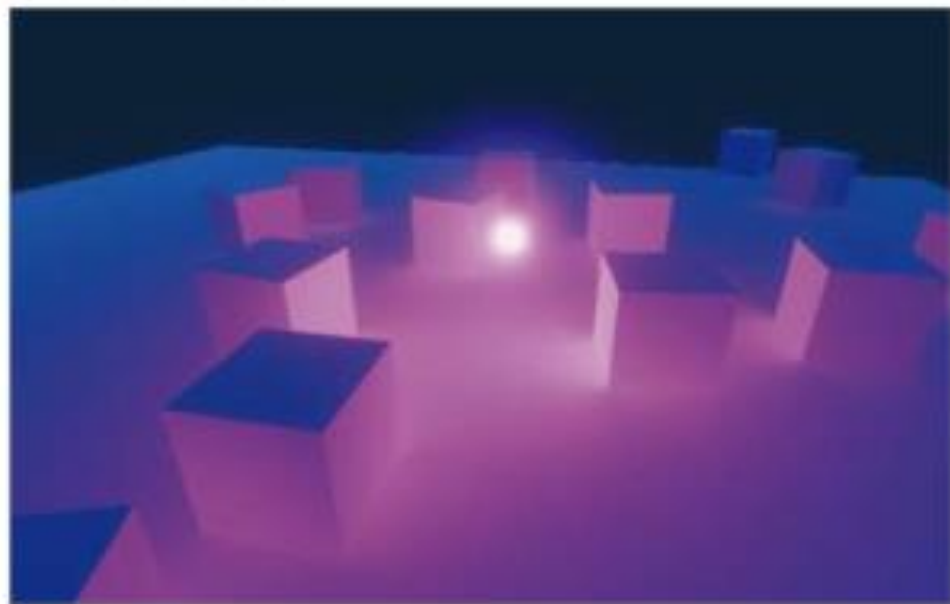


레벨과 라이트맵

GI를 사용하지 않은 경우



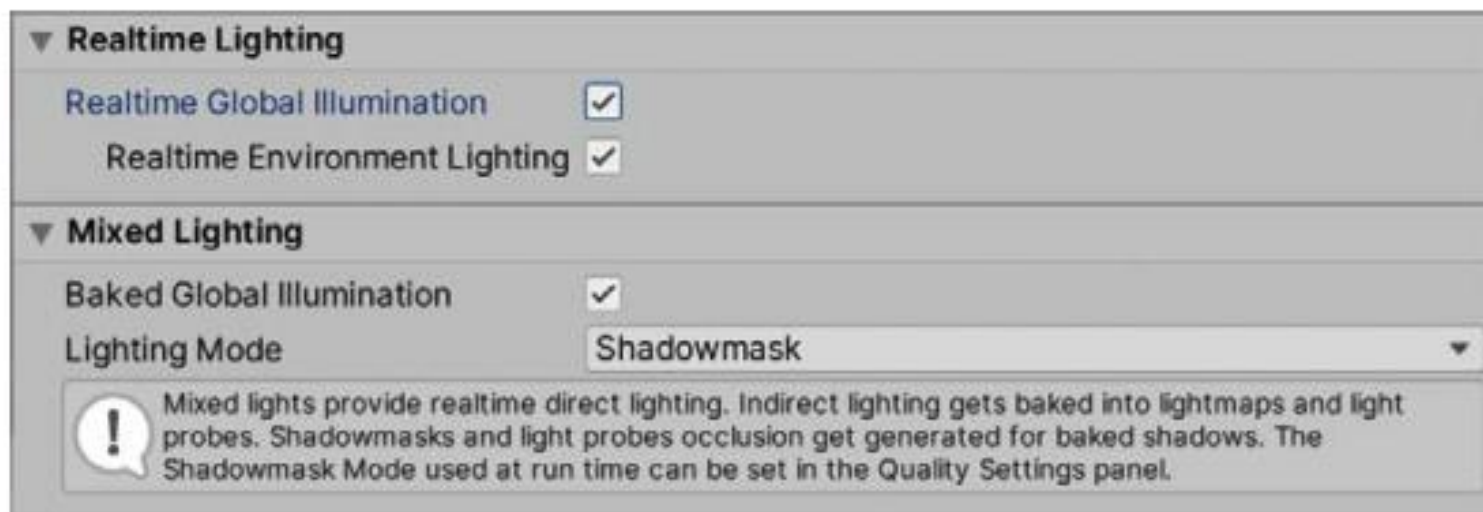
GI를 사용한 경우



▶ 글로벌 일루미네이션의 적용 여부에 따른 차이

레벨과 라이트맵

- 실시간 글로벌 일루미네이션(Realtime Global Illumination)
- 베이킹된 글로벌 일루미네이션(Baked Global Illumination)



▶ 글로벌 일루미네이션 설정

레벨과 라이트맵

- 동적 게임 오브젝트

실시간 직사광, 실시간 그림자, 그림자 맵을 활용한 실시간 그림자

- 정적 게임 오브젝트

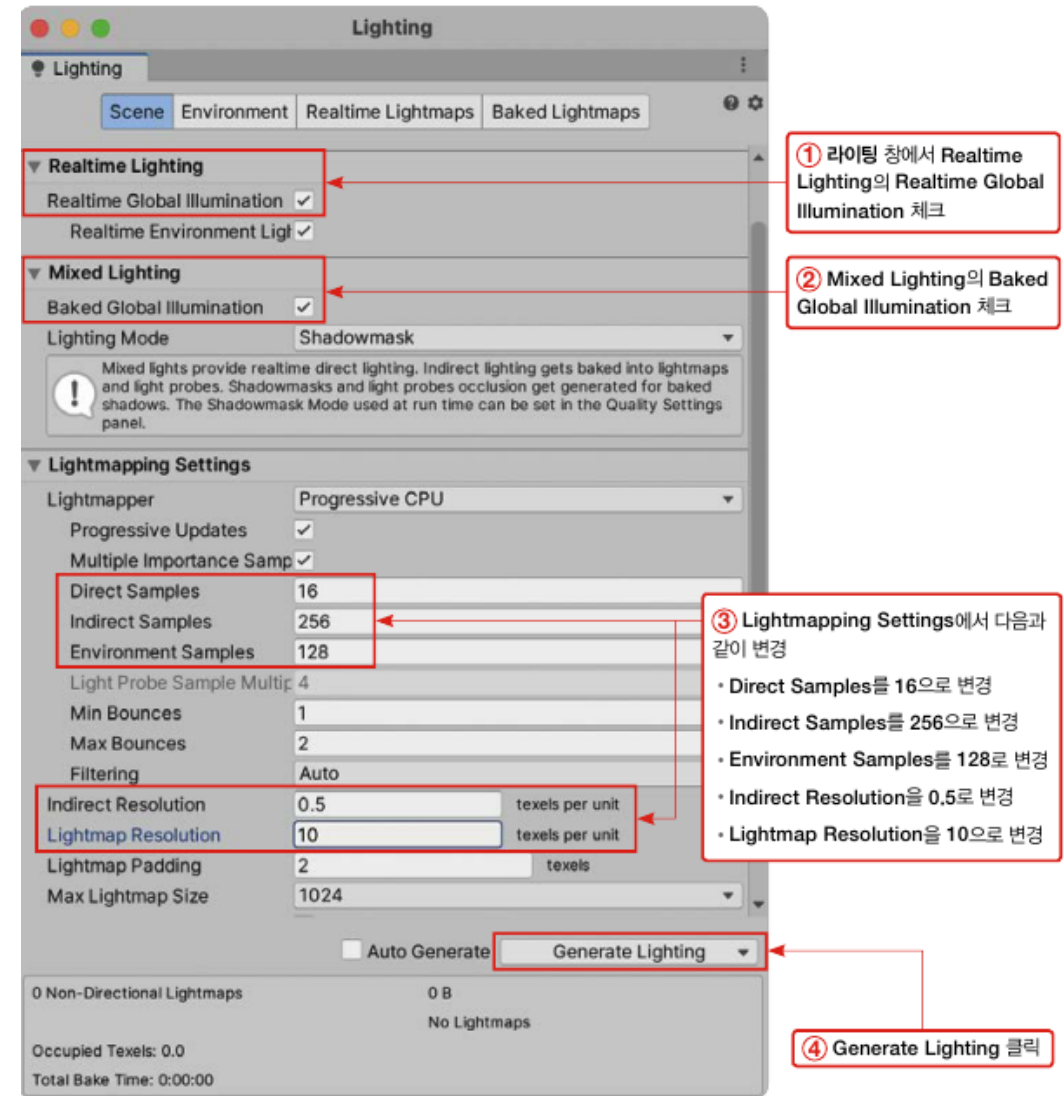
실시간 직사광, 라이트맵을 통해 미리 구워진 간접광, 미리 구워진 그림자, 그림자 맵을 활용한 실시간 그림자

게임오브젝트의 인스펙터에는 맨 우측상단에 Static 체크박스
해당 오브젝트가 움직이지 않을 것이라고 알려주기 위해 사용
Static 체크 후에도 에디터에서 오브젝트 변형 가능함
게임 도중에는 변형 불가(스크립트로 가능)

레벨과 라이트맵

[과정 1] 라이트맵 굽기

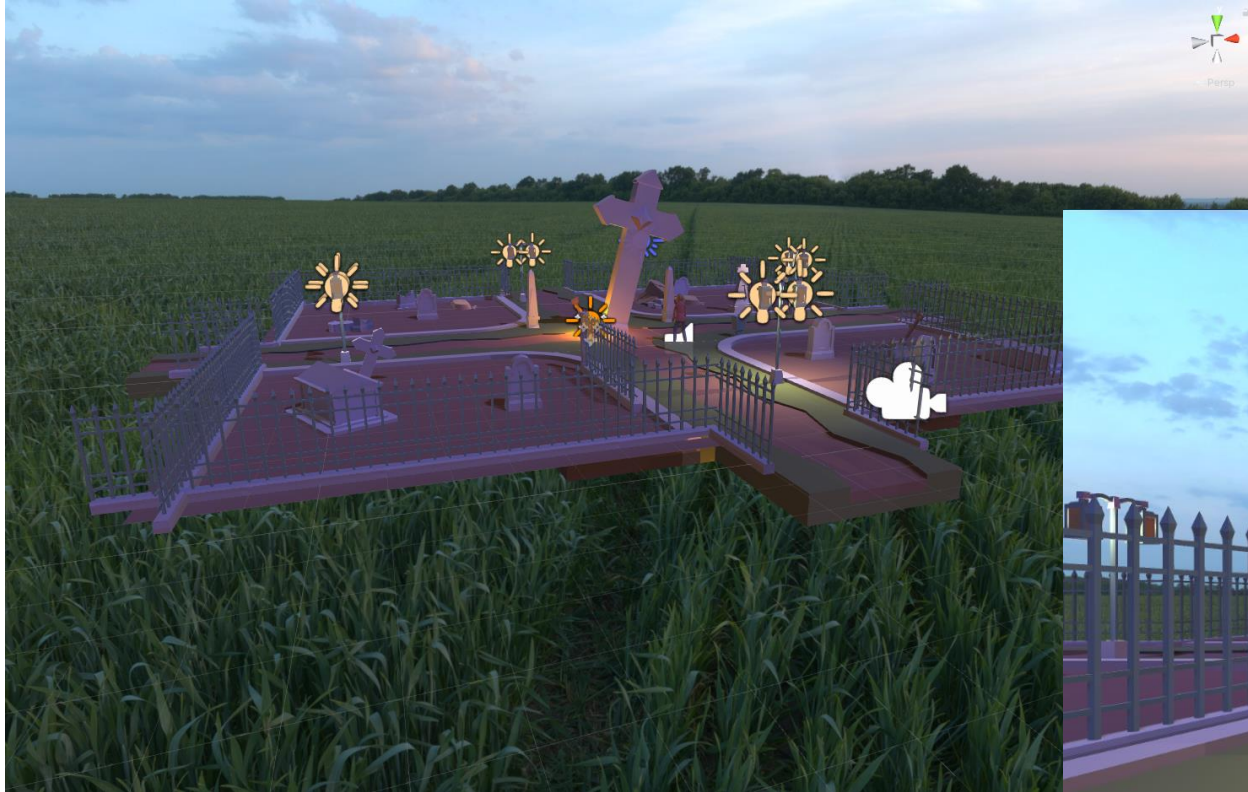
- ① 라이팅 창에서 Realtime Lighting의 Realtime Global Illumination 체크
- ② Mixed Lighting의 Baked Global Illumination 체크
- ③ Lightmapping Settings에서 다음과 같이 변경
 - Direct Samples를 16으로 변경
 - Indirect Samples를 256으로 변경
 - Environment Samples를 128로 변경
 - Indirect Resolution을 0.5로 변경
 - Lightmap Resolution을 10으로 변경
- ④ Generate Lighting 클릭



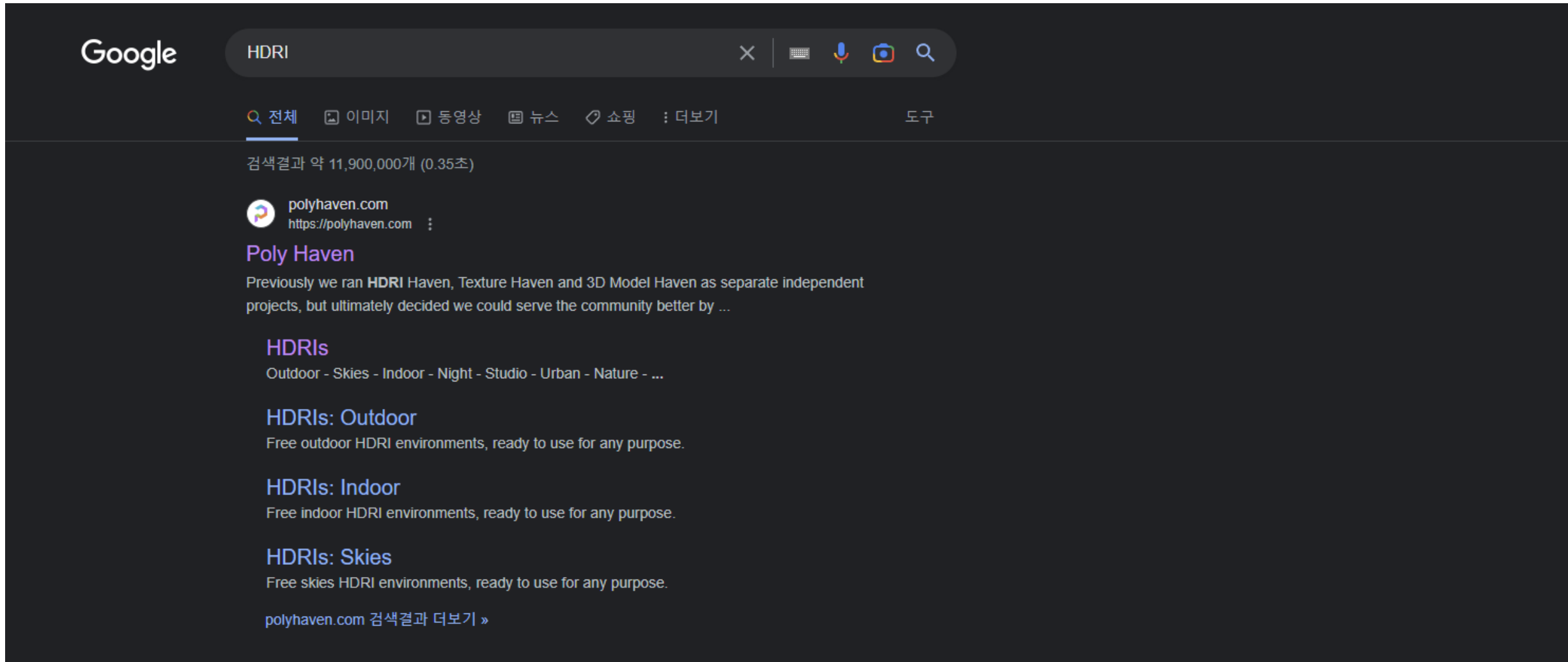
▶ 라이트맵 굽기

레벨과 라이트맵

- Sky Box



레벨과 라이트맵




The screenshot shows a Google search interface with the query 'HDRI'. The search results page displays the top result for 'polyhaven.com'. The page content includes a description of the site, which previously ran HDRI Haven, Texture Haven, and 3D Model Haven, and now offers a unified resource. It lists categories of HDRI environments: Outdoor, Indoor, and Skies, each with a brief description of the available content.

Google

HDRI

전체 이미지 동영상 뉴스 쇼핑 더보기 도구

검색결과 약 11,900,000개 (0.35초)

 polyhaven.com
https://polyhaven.com

Poly Haven

Previously we ran **HDRI Haven**, **Texture Haven** and **3D Model Haven** as separate independent projects, but ultimately decided we could serve the community better by ...

HDRIs
Outdoor - Skies - Indoor - Night - Studio - Urban - Nature - ...


HDRIs: Outdoor
Free outdoor HDRI environments, ready to use for any purpose.

HDRIs: Indoor
Free indoor HDRI environments, ready to use for any purpose.

HDRIs: Skies
Free skies HDRI environments, ready to use for any purpose.

polyhaven.com 검색결과 더보기 »

레벨과 라이트맵



Poly Haven
The Public 3D Asset Library

[Assets](#) [Add-on](#) [Gallery](#)


[Add-on](#) [Gallery](#) [Support Us](#) [About/Contact](#) [🇬🇧](#)

4K
▼


HDR
▼

Download
17.83 MB

☰

**Jarod Guest**
Sky Edits
✉

Authors:

**Sergej Majboroda**
Original
🔗 ✉


Edited sky-only version of [Evening Road 01](#).

CC0
License

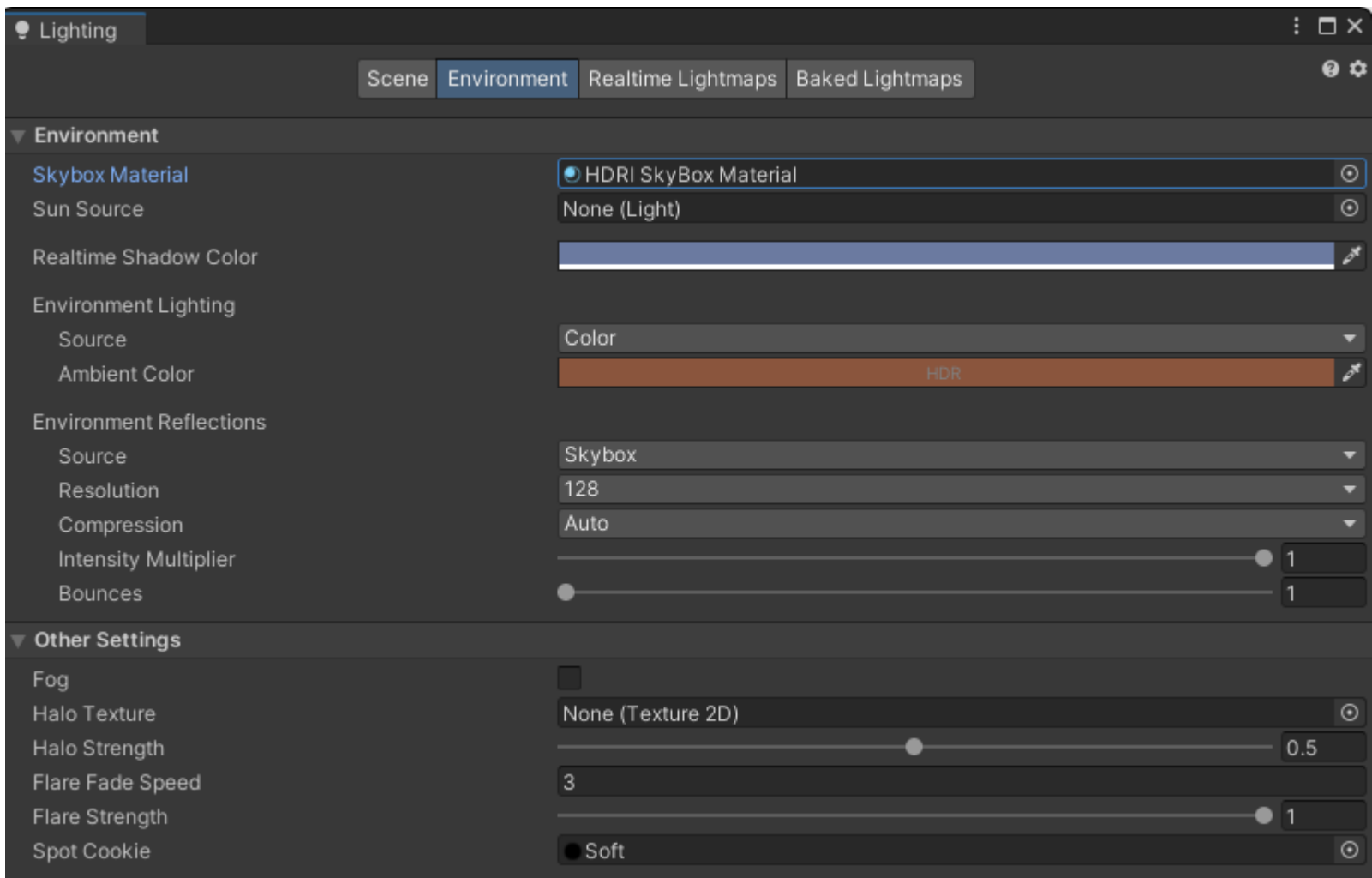
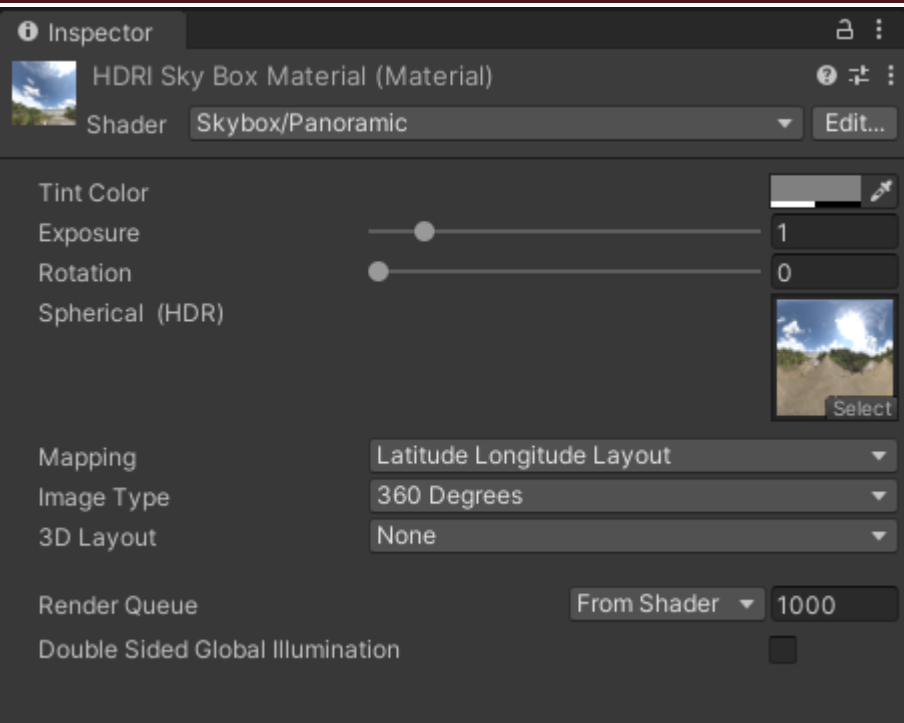
1
month ago

12
EVs

5400K
WB

Downloads: 35566 

레벨과 라이트맵



레벨과 라이트맵

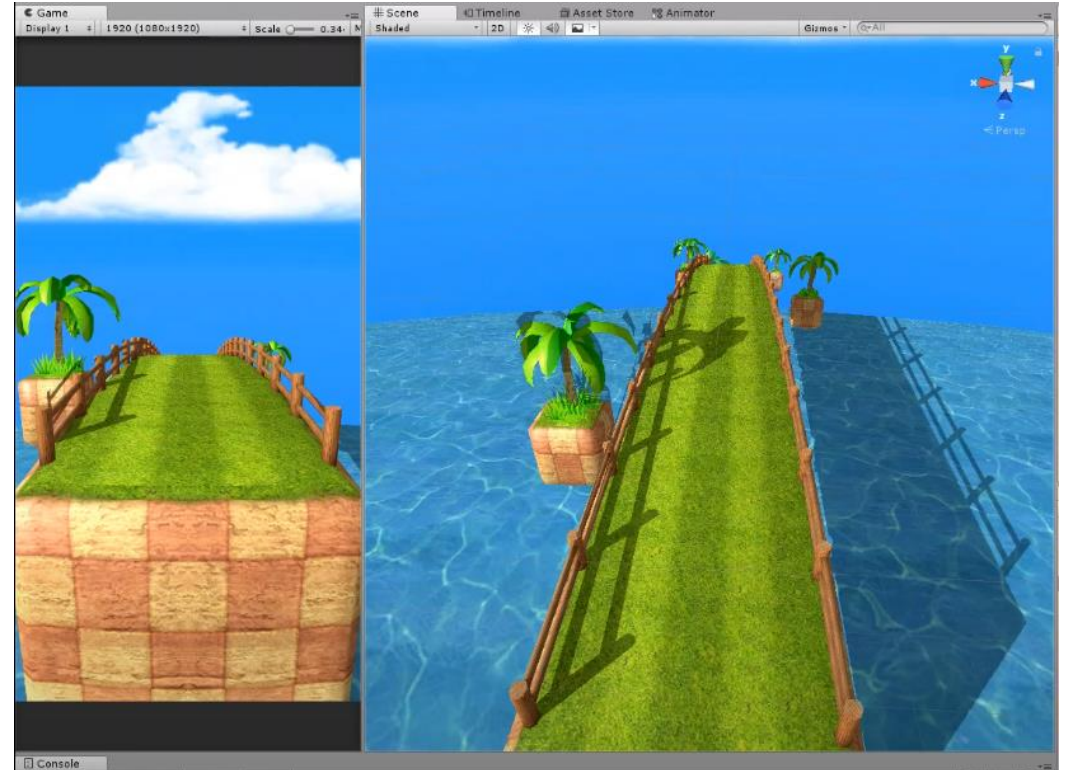
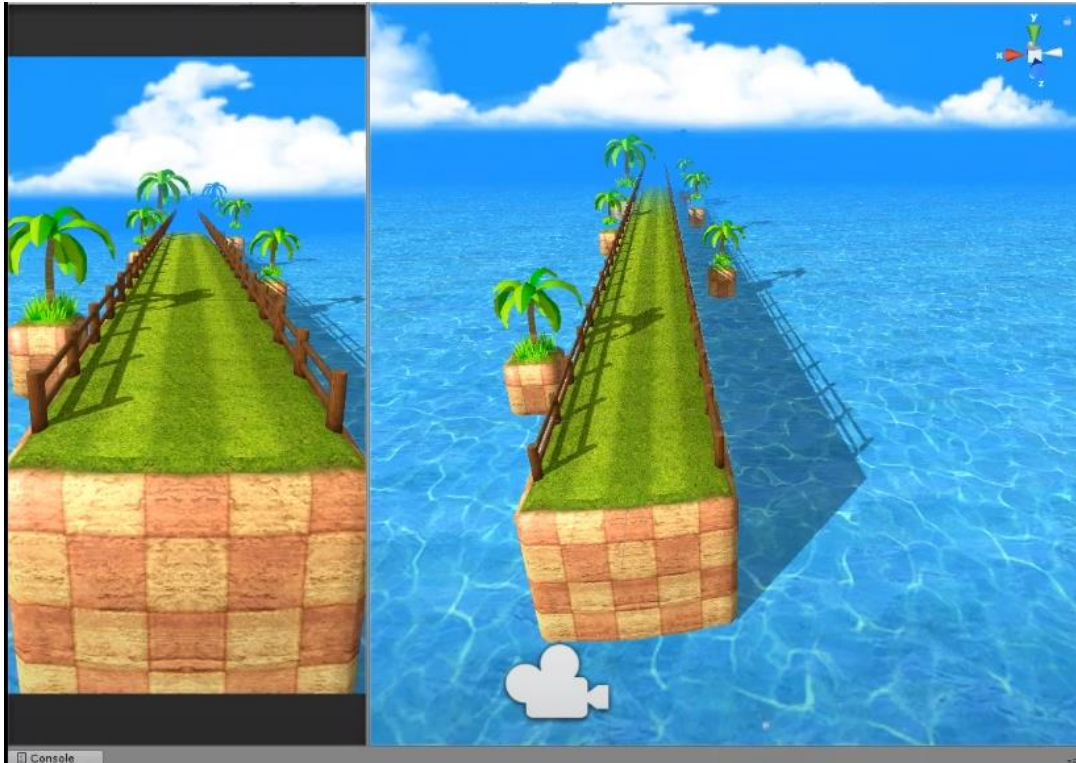
- 360 VR
InnerSphere



레벨과 라이트맵

- Curve The World

Shader를 이용 하여 Curved World를 구현



플레이어 캐릭터와 애니메이션

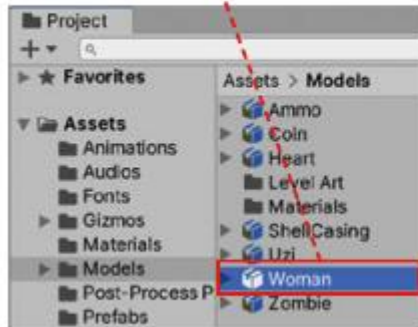
플레이어 캐릭터와 애니메이션

[과정 01] 캐릭터 추가하기

- ① Models 폴더에서 Woman 모델을 하이얼라키 창으로 드래그&드롭
- ② 생성된 Woman 게임 오브젝트의 이름을 Player Character로, 태그를 Player로 변경
- ③ 위치를 (0, 0, 0)으로 변경

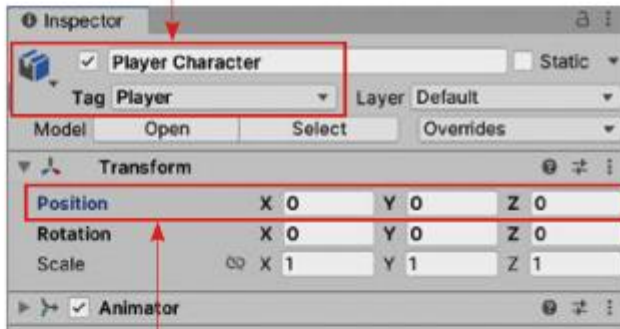


- ① Woman 모델을 하이얼라키 창으로 드래그&드롭



▶ 캐릭터 추가하기

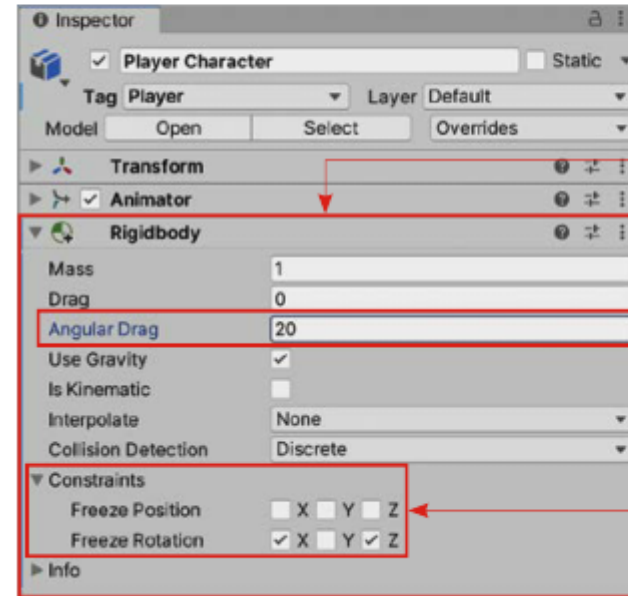
- ② 이름을 Player Character로, 태그를 Player로 변경



- ③ 위치를 (0, 0, 0)으로 변경

[과정 02] 캐릭터에 리지드바디 추가하기

- ① Rigidbody 컴포넌트 추가(Add Component > Physics > Rigidbody)
- ② Rigidbody 컴포넌트의 Angular Drag를 20으로 변경
- ③ Rigidbody 컴포넌트의 Constraints 탭 펼치기 > Freeze Rotation X와 Z 체크



- ① Rigidbody 컴포넌트 추가

- ② Angular Drag를 20으로 변경

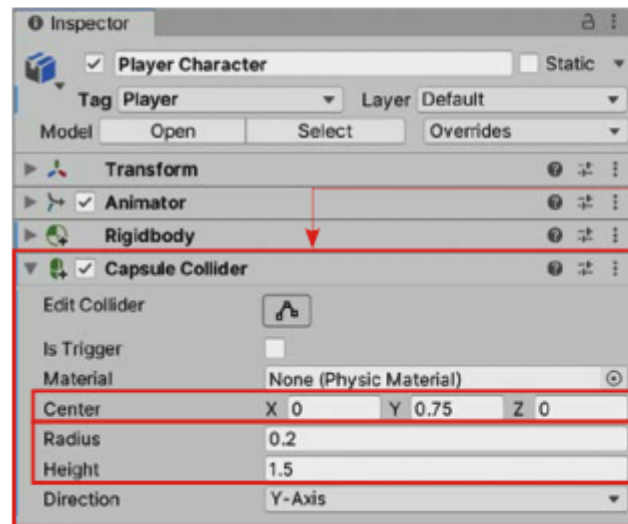
- ③ Constraints 탭 펼치기 > Freeze Rotation X와 Z 체크

▶ 캐릭터에 리지드바디 컴포넌트 추가하기

플레이어 캐릭터와 애니메이션

[과정 03] 캐릭터에 캡슐 콜라이더 추가하기

- ① Capsule Collider 컴포넌트 추가(Add Component > Physics > Capsule Collider)
- ② Capsule Collider 컴포넌트의 Center를 (0, 0.75, 0)으로 변경
- ③ Capsule Collider 컴포넌트의 Radius를 0.2, Height를 1.5로 변경



① Capsule Collider 컴포넌트 추가

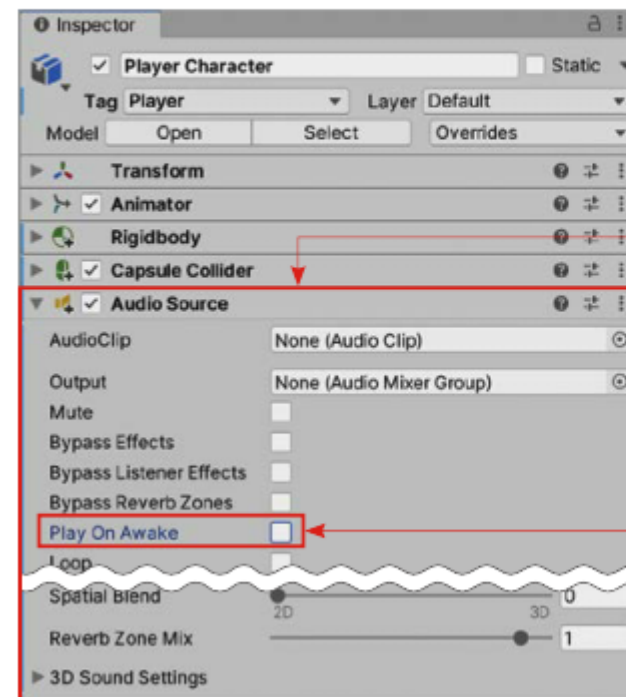
② Center를 (0, 0.75, 0)으로 변경

③ Radius를 0.2, Height를 1.5로 변경

▶ 캐릭터에 캡슐 콜라이더 추가하기

[과정 04] 캐릭터에 오디오 소스 추가하기

- ① Audio Source 컴포넌트 추가(Add Component > Audio > Audio Source)
- ② Audio Source 컴포넌트의 Play On Awake 체크 해제



① Audio Source 컴포넌트 추가

② Play On Awake 체크 해제

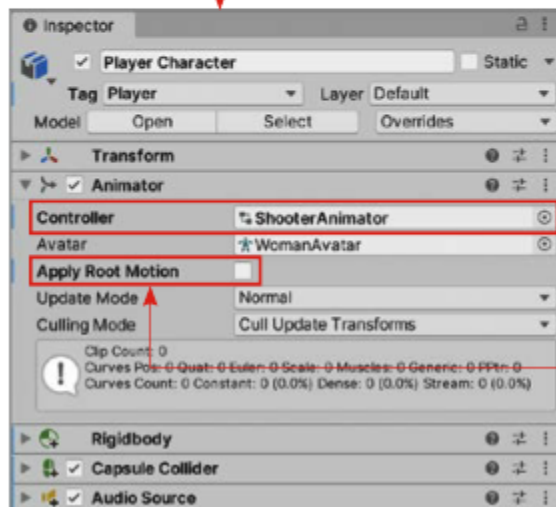
▶ 캐릭터에 오디오 소스 추가하기

플레이어 캐릭터와 애니메이션

[과정 이] 캐릭터의 애니메이터 설정하기

- ① 하이어라키 창에서 Player Character 게임 오브젝트 선택
- ② Animator 컴포넌트의 Controller 필드에 ShooterAnimator 애니메이터 컨트롤러 할당
(Controller 필드의 선택 버튼 클릭 > 선택 창에서 ShooterAnimator 더블 클릭)
- ③ Animator 컴포넌트의 Apply Root Motion 체크 해제

① Player Character 게임 오브젝트 선택



② Controller 필드에 ShooterAnimator
애니메이터 컨트롤러 할당

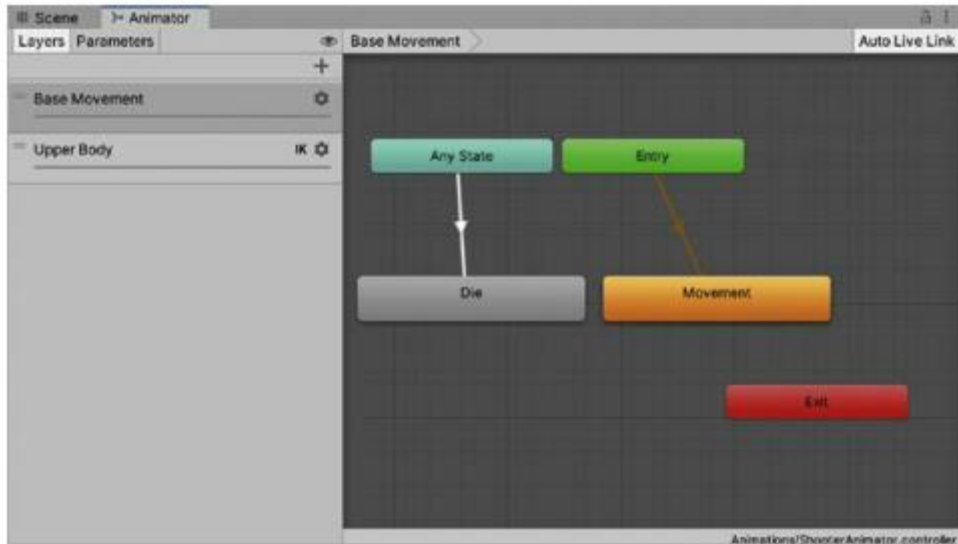
③ Apply Root Motion 체크 해제

▶ 캐릭터의 애니메이터 설정하기

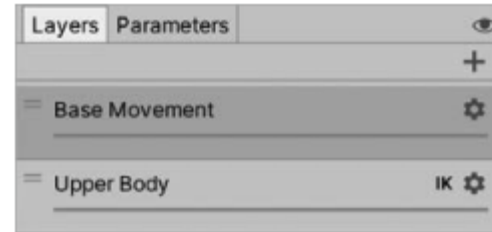
플레이어 캐릭터와 애니메이션

[과정 1] ShooterAnimator 애니메이터 컨트롤러 표시하기

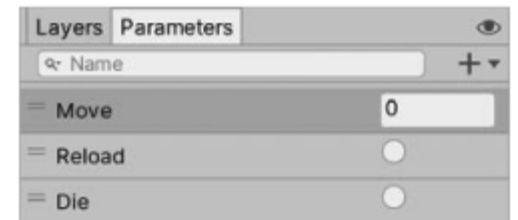
- ① 하이어라키 창에서 **Player Character** 게임 오브젝트 선택
- ② 애니메이터 창 띄우기(유니티 상단 메뉴의 **Window > Animation > Animator**)



▶ 열린 애니메이터 창



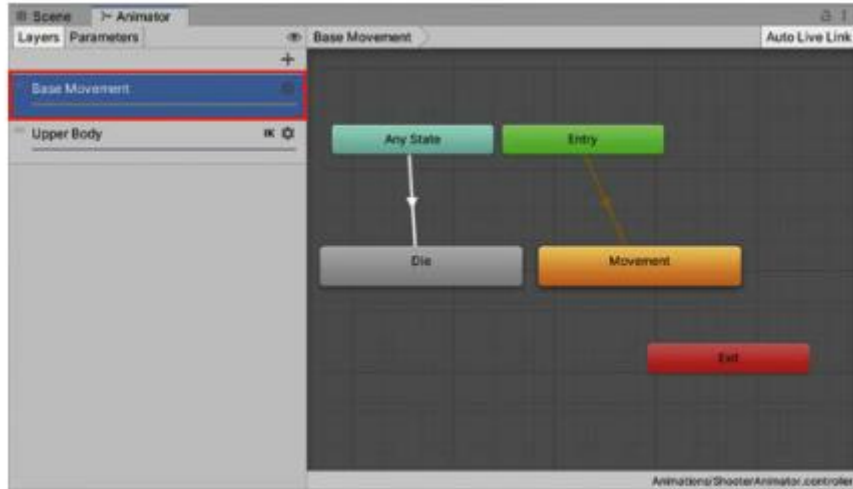
▶ 두 레이어



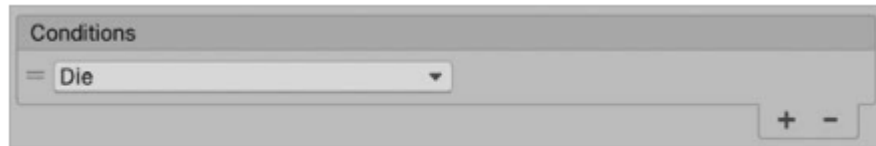
▶ 파라미터 리스트

- Move : 앞뒤 움직임에 관한 입력값
- Reload : 재장전을 알리는 트리거
- Die : 사망을 알리는 트리거

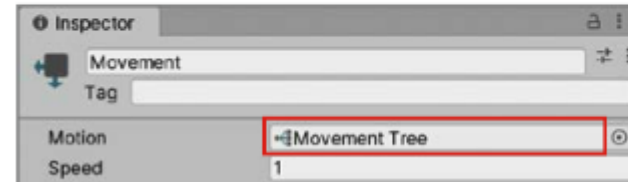
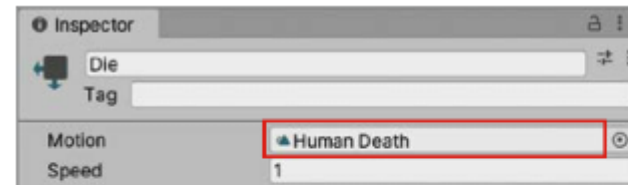
플레이어 캐릭터와 애니메이션



▶ Base Movement 레이어

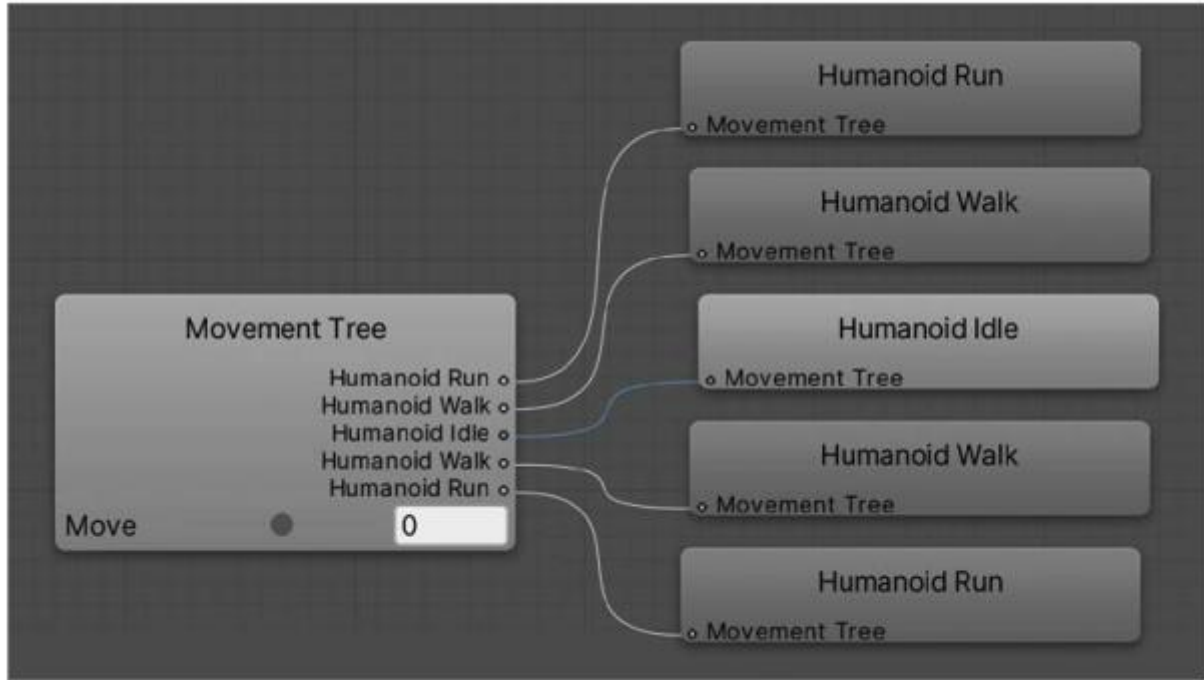


▶ Any State → Die 전이의 조건

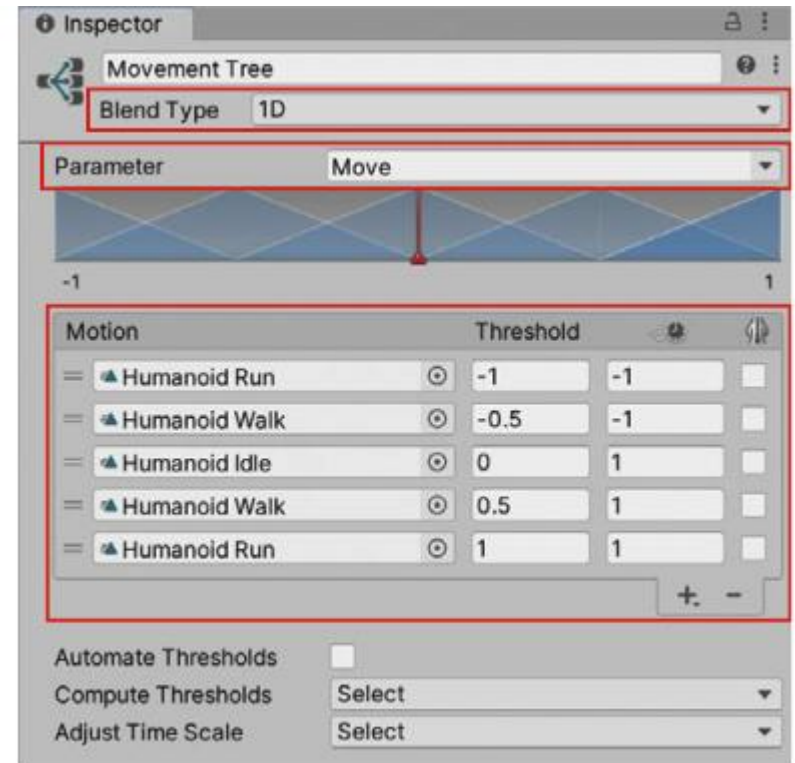


▶ Die 상태에는 애니메이션 클립, Movement 상태에는 블렌드 트리가 할당됨

플레이어 캐릭터와 애니메이션



▶ Movement 상태의 블렌드 트리 그래프



▶ Movement 상태의 Blend Tree 모션 구성

플레이어 캐릭터와 애니메이션

순서	애니메이션 클립(Motion)	임계값(Threshold)	애니메이션 재생속도(Animation Speed)
1	Humanoid Run(뒤로 뛰기)	-1	-1
2	Humanoid Walk(뒤로 걷기)	-0.5	-1
3	Humanoid Idle(대기)	0	1
4	Humanoid Walk(걷기)	0.5	1
5	Humanoid Run(뛰기)	1	1

플레이어 캐릭터와 애니메이션

- 블렌드 트리

여러 개의 애니메이션을 블렌드하고, 각 애니메이션의 일부가 각각 다른 비중으로 합쳐져 매끄럽게 블렌딩되도록 하기 위해 사용

각각의 모션이 최종 효과에 어느 정도 영향을 주도록 할 것인지는 블렌딩 파라미터 를 사용하여 제어

블렌드 트리는 애니메이션 상태 머신의 특별한 스테이트 타입

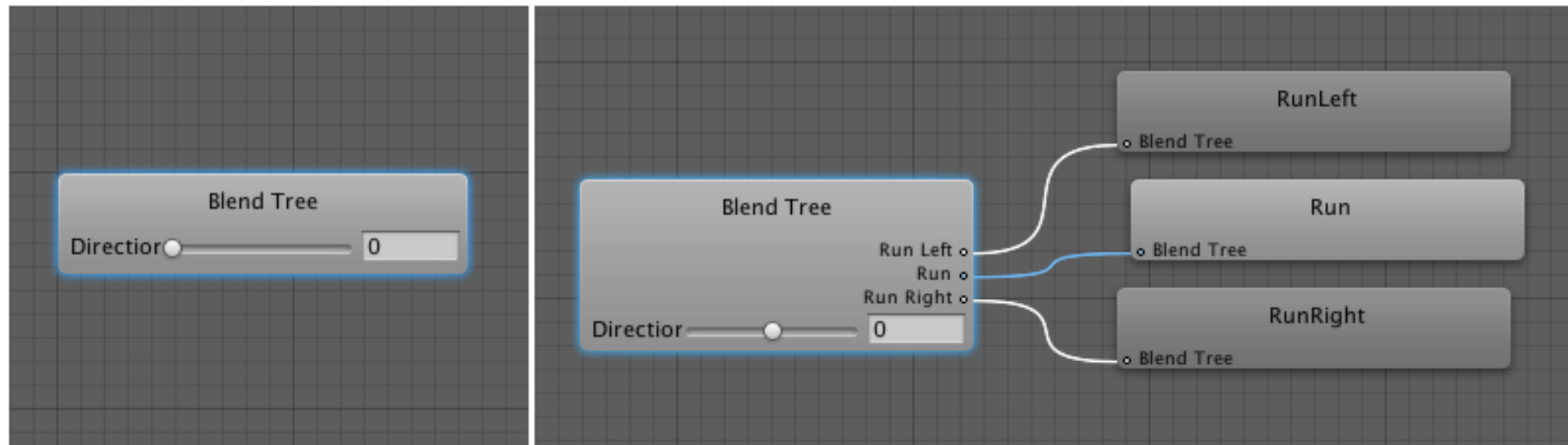
플레이어 캐릭터와 애니메이션

블렌드 트리 사용

새로운 블렌드 트리에서 작업하기 위해 다음 단계를 수행합니다.

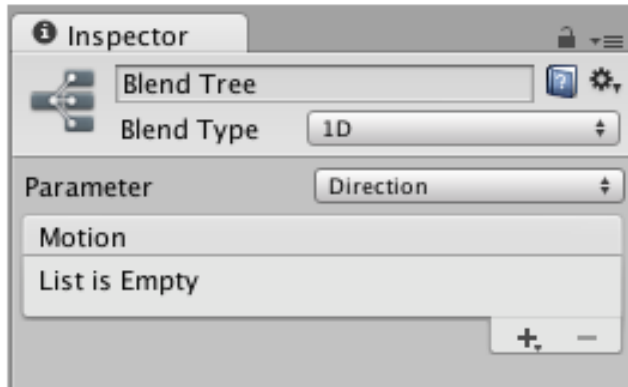
1. 애니메이터 컨트롤러 창의 빈 공간에 마우스를 마우스 오른쪽 버튼으로 클릭합니다.
2. 컨텍스트 메뉴가 나타나면 **Create State > From New Blend Tree** 를 선택합니다.
3. 블렌드 트리를 더블 클릭하여 블렌드 트리 그래프를 엽니다.

이제 애니메이터 창에 전체 블렌드 트리의 그래프가 나타나며, 인스펙터는 현재 선택된 노드와 그 직계 자식 노드를 보여줍니다.



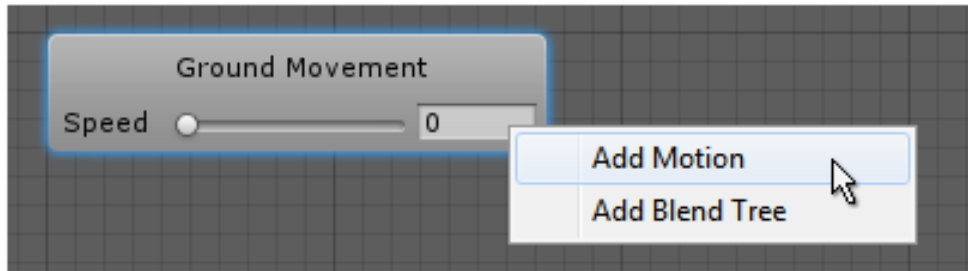
플레이어 캐릭터와 애니메이션

애니메이션 클립을 블렌드 트리에 추가하기 위해서는 블렌드 트리를 선택하고 인스펙터의 모션 필드에 있는 + 아이콘을 클릭하면 됩니다.



모션이 추가되지 않은 상태의 블렌드 노드를 인스펙터에서 볼 경우, + 아이콘을 사용하여 애니메이션 클립이나 자식 블렌드 트리를 추가합니다.

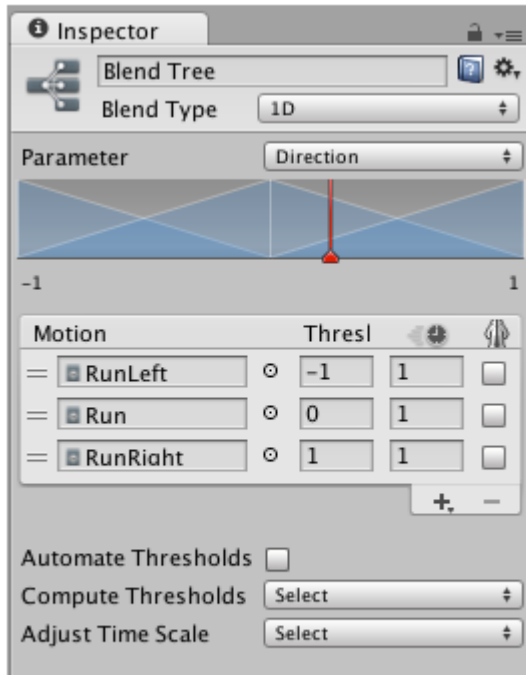
또는 블렌드 트리에서 마우스 오른쪽 버튼으로 클릭하고 컨텍스트 메뉴에서 선택하는 방식으로 애니메이션 클립이나 자식 블렌드 노드를 추가할 수도 있습니다.



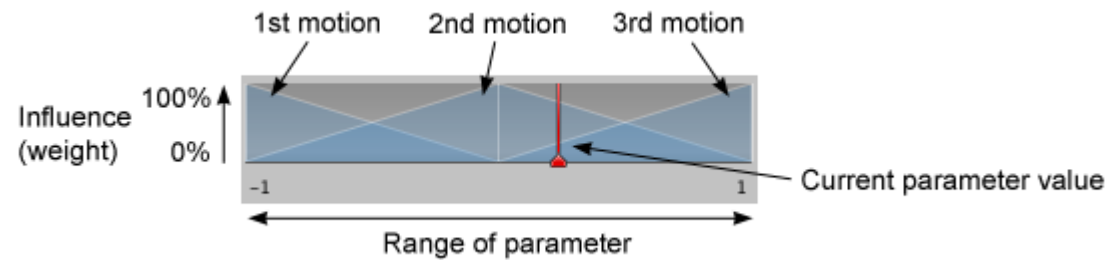
블렌드 트리 노드에서 마우스 오른쪽 버튼으로 클릭했을 때의 컨텍스트 메뉴

플레이어 캐릭터와 애니메이션

- 1D 블렌딩



세 애니메이션 클립을 포함한 1D 블렌드 트리.



플레이어 캐릭터와 애니메이션

- 2D 블렌딩

- 2D Simple Directional

모션이 “앞으로 걷기”, “뒤로 걷기”, “왼쪽으로 걷기”, “오른쪽으로 걷기”, 또는 “위로 겨냥”, “아래로 겨냥”, “왼쪽으로 겨냥”, “오른쪽으로 겨냥” 같은 다른 방향을 가리킬 때 가장 유용.

- 2D Freeform Directional

이 블렌드 타입도 모션이 다른 방향을 표시할 때 사용되지만, “앞으로 걷기”, “앞으로 뛰기”와 같이 같은 방향에 여러 모션을 가질 수 있음.

프리폼 디렉셔널 타입에서 모션 설정은 “대기 상태”와 같은 포지션 (0, 0)에 싱글 모션을 항상 포함

- 2D Freeform Cartesian

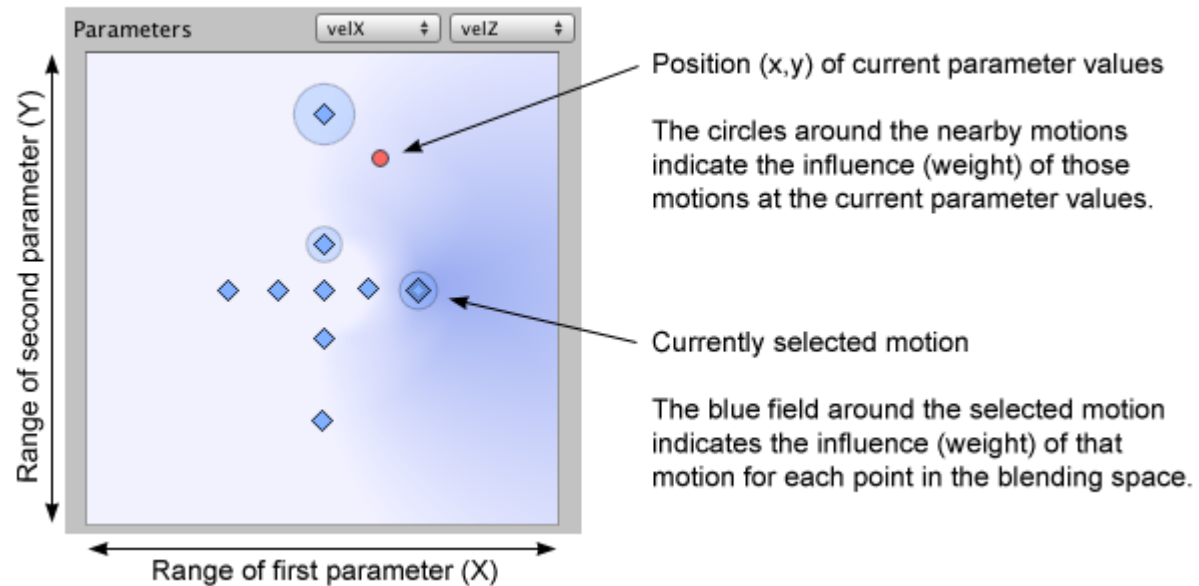
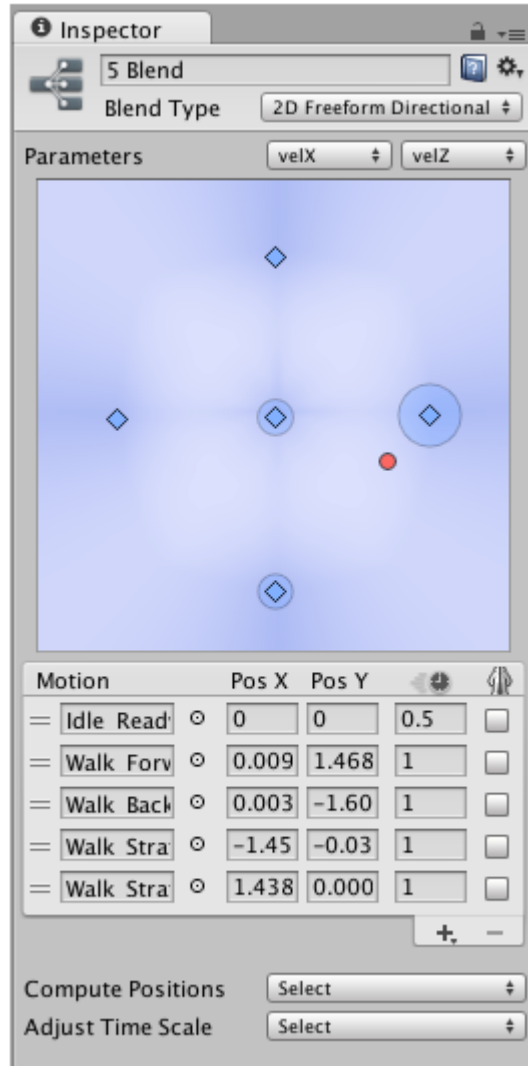
모션이 다른 방향을 가리키지 않을 때 가장 유용.

프리폼 카르테지안에서 X 파라미터와 Y 파라미터는 각속도와 리니어 속도와 같이 다른 컨셉을 가리킬 수 있음.
(예 “앞으로 걷고 돌지 않기”, “앞으로 뛰고 돌지 않기”, “앞으로 걷고 오른쪽으로 돌기”, “앞으로 뛰고 오른쪽으로 돌기” 등과 같은 모션)

- Direct

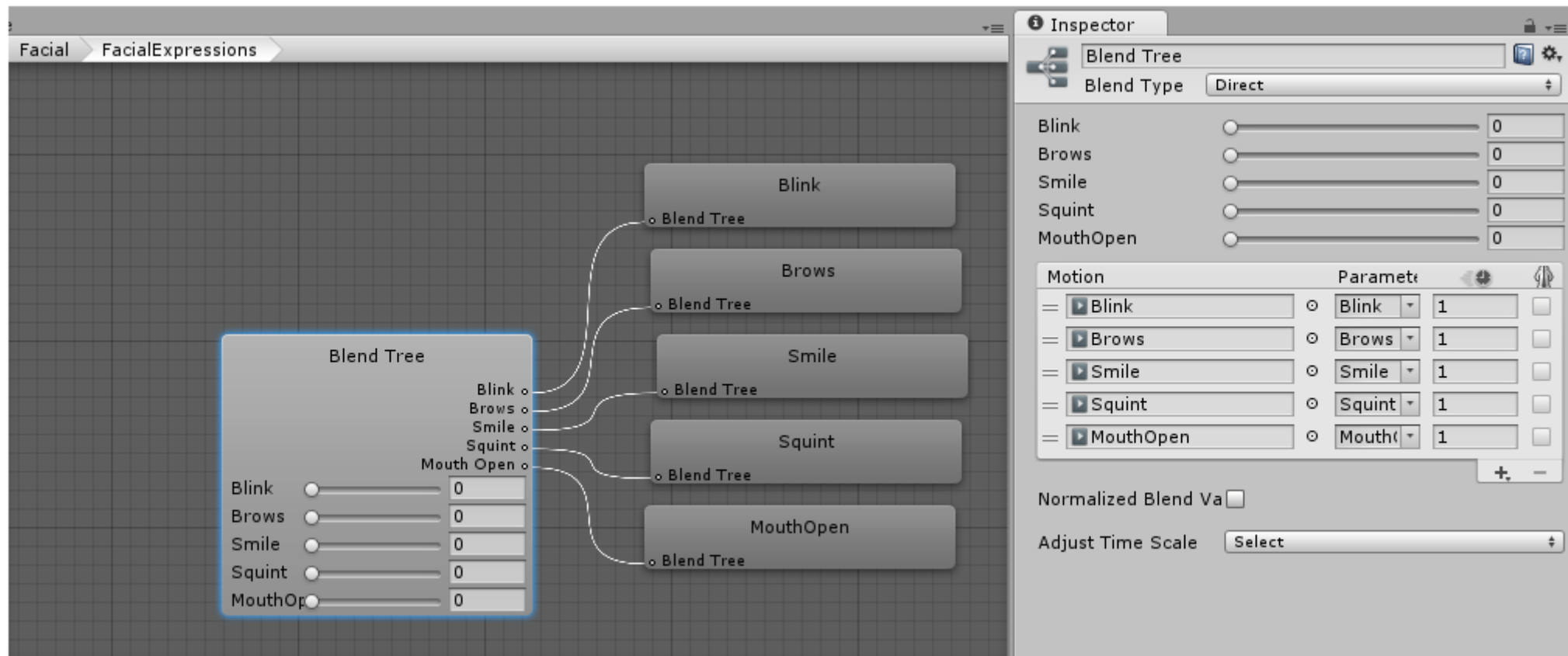
블렌드 트리의 타입은 각 노드의 가중치를 직접 컨트롤.
얼굴 표정이나 랜덤한 대기 상태의 블렌딩에 유용.

플레이어 캐릭터와 애니메이션



플레이어 캐릭터와 애니메이션

- 다이렉트 블렌딩



플레이어 캐릭터와 애니메이션

• 다이렉트 블렌딩

자식 BlendTree의 가중치에 대해 애니메이터 파라미터를 직접적으로 매핑할 수 있게 함.

이것은, 하나 또는 두 개의 파라미터를 사용하여 간접적으로 블렌드시키는 것보다 블렌드되고 있는 다양한 애니메이션에 대한 완전한 제어를 원할 때 유용하게 사용할 수 있음(1D와 2D 블렌드 트리의 경우).

다이렉트 블렌드 트리를 설정할 때 인스펙터를 통해 모션 리스트에 모션을 더할 수 있습니다. 각 모션은 이후 트리에서의 블렌드 가중치를 직접적으로 제어하기 위해 부합하는 파라미터에 할당되어야 합니다. 여기에서 애니메이터 파라미터를 만드는 방법에 대해 더 자세히 알아볼 수 있습니다.

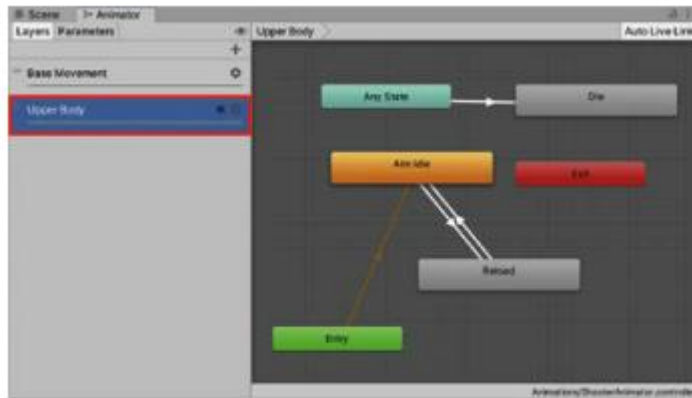
크로스페이딩이나 다양한 2D 블렌딩 알고리즘을 건너뛸(프리폼 디렉셔널, 프리폼 카테시안, 기타).

사용자가 블렌드된 혼합 애니메이션에 대해 어떤 코드든지 조절.

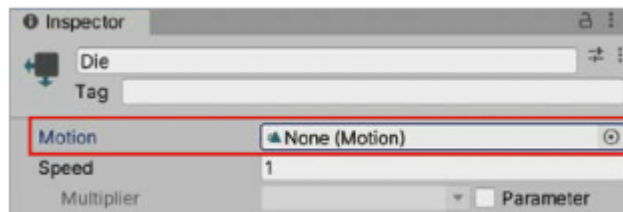
블렌드 셰이프 애니메이션을 섞어 얼굴 표정을 만들거나 추가 애니메이션을 서로 블렌딩할 때 특히 유용.



플레이어 캐릭터와 애니메이션



▶ Upper Body 레이어의 구성

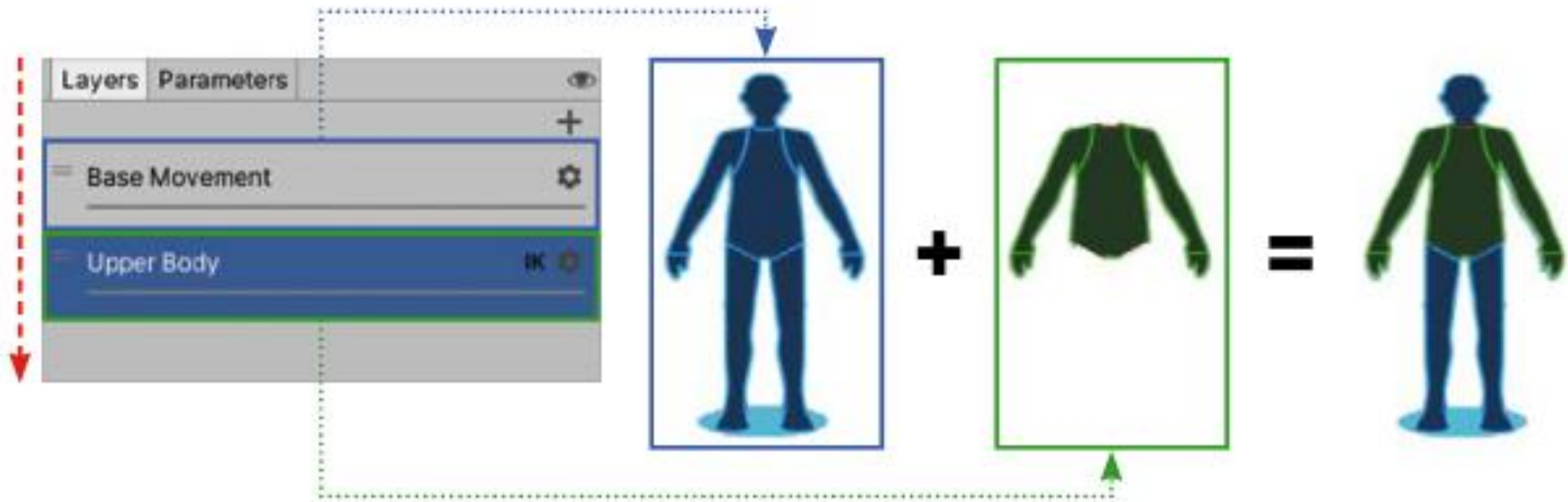


▶ Motion 필드가 빈 Die 상태



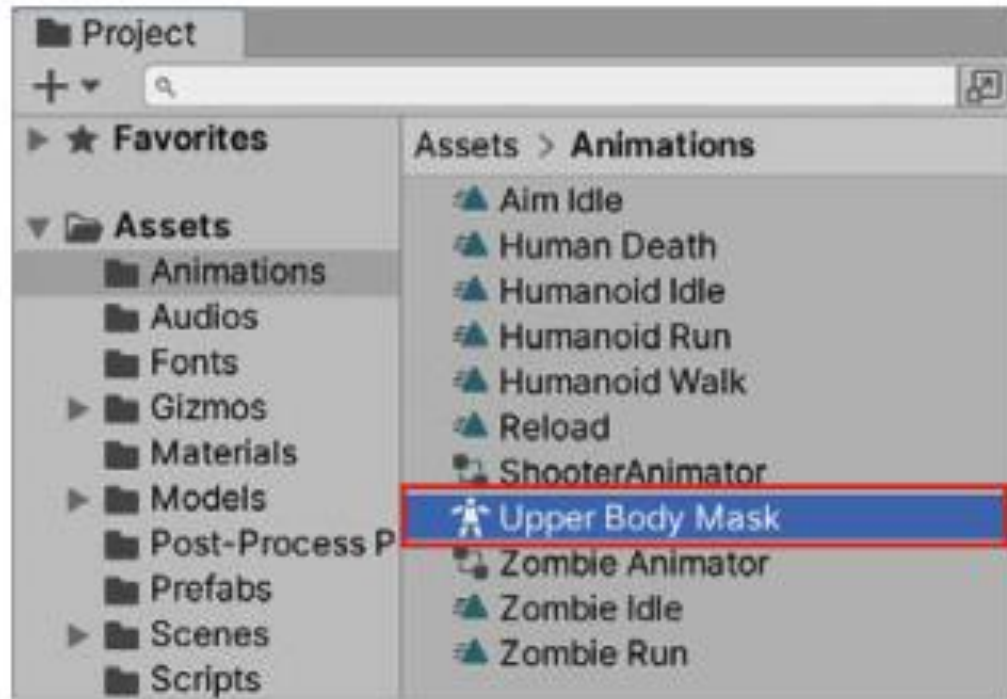
▶ Aim Idle 상태와 Reload 상태에 할당된 애니메이션 클립

플레이어 캐릭터와 애니메이션



▶ Base Movement 레이어와 Upper Body 레이어의 혼합

플레이어 캐릭터와 애니메이션

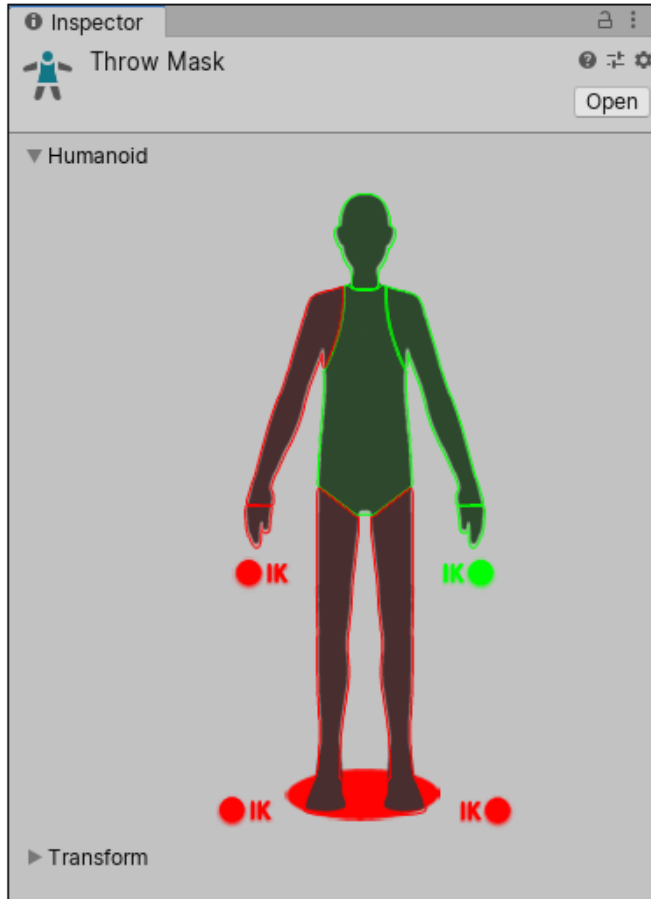


▶ Upper Body Mask 아바타 마스크

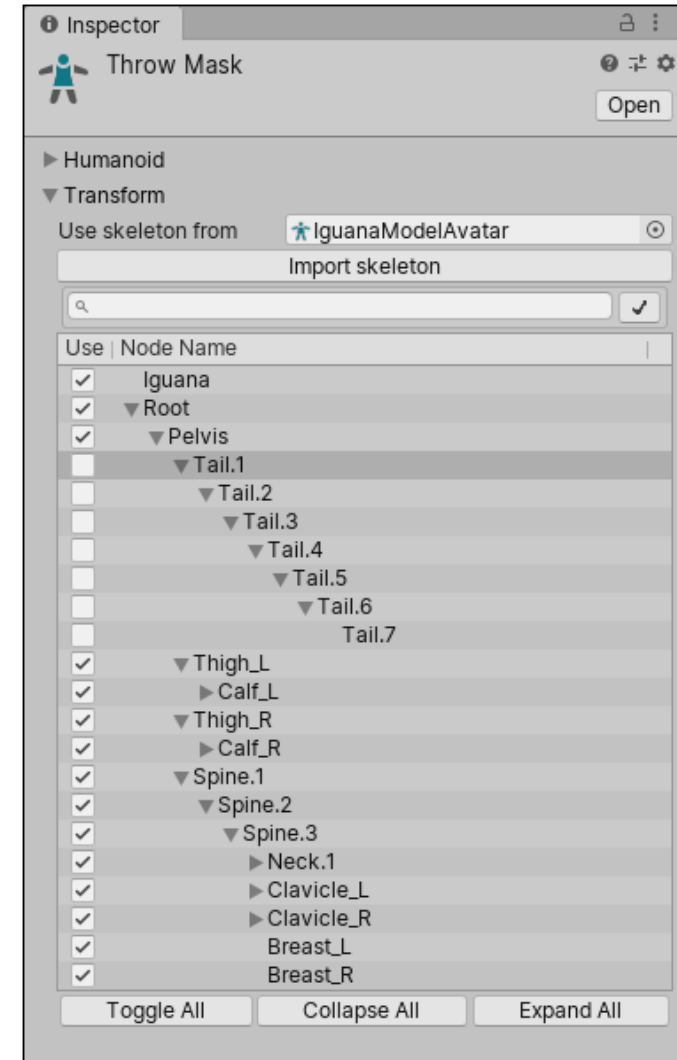


플레이어 캐릭터와 애니메이션

- 아바타 마스크



휴머노이드 바디를 사용하여 아바타 마스크 정의



트랜스폼 메서드를 사용한 아바타 마스크 정의

플레이어 캐릭터와 애니메이션

- 역운동학(IK)

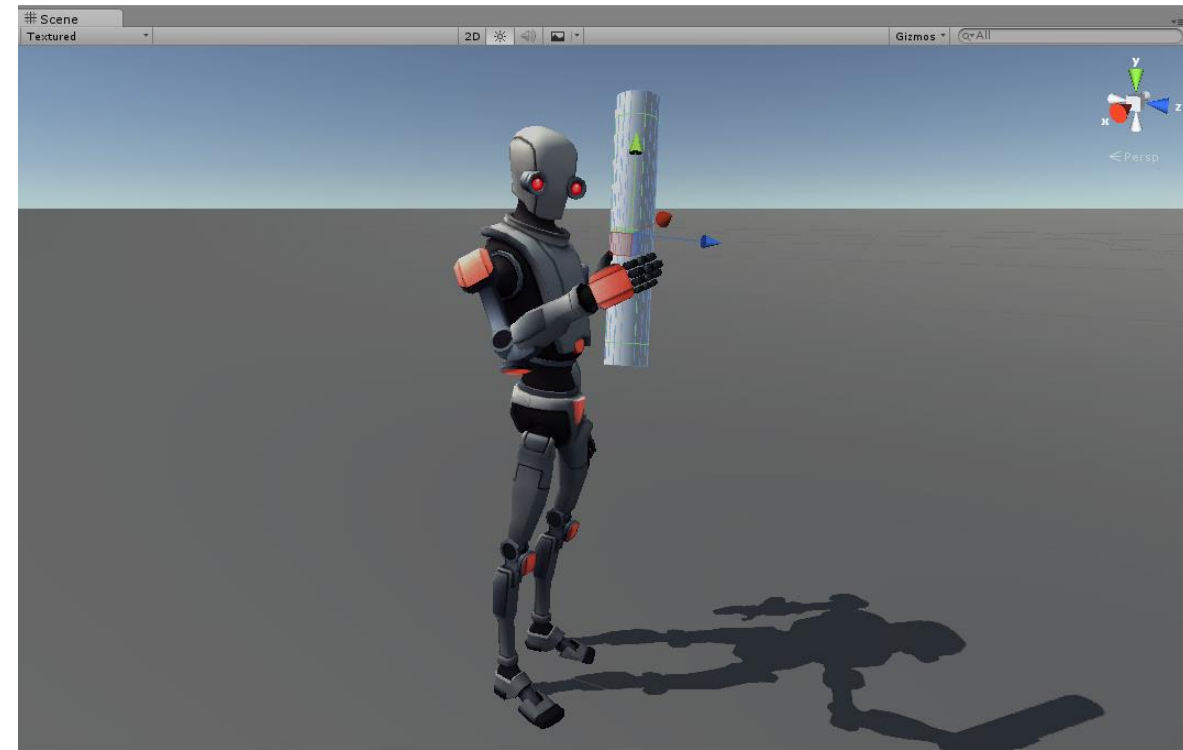
대부분의 애니메이션은 스켈레톤의 조인트 각도를 미리 정해진 값으로 회전하여 만듦. 자식 조인트의 위치는 부모의 회전에 따라 변하므로 조인트 체인의 끝 점은 체인에 포함된 각 조인트의 각도와 상대 위치에 따라 결정. 이런 스켈레톤 포즈 메서드를 순운동학(FK).

하지만 조인트 포즈 작업을 반대 시각에서 바라보는 것이 유용한 경우도 많음.

공간에서 선택된 위치에 따라서는 역으로 작업하여 적합한 조인트 방향을 찾아 해당 위치에 조인트 끝점이 오도록 하는 방법이 유용.

이 방법은 사용자가 선택한 포인트의 오브젝트를 캐릭터가 건드리게 하거나, 울퉁불퉁한 표면 위에 캐릭터의 두 발이 자연스럽게 밀착해있도록 하려는 경우에 유용.

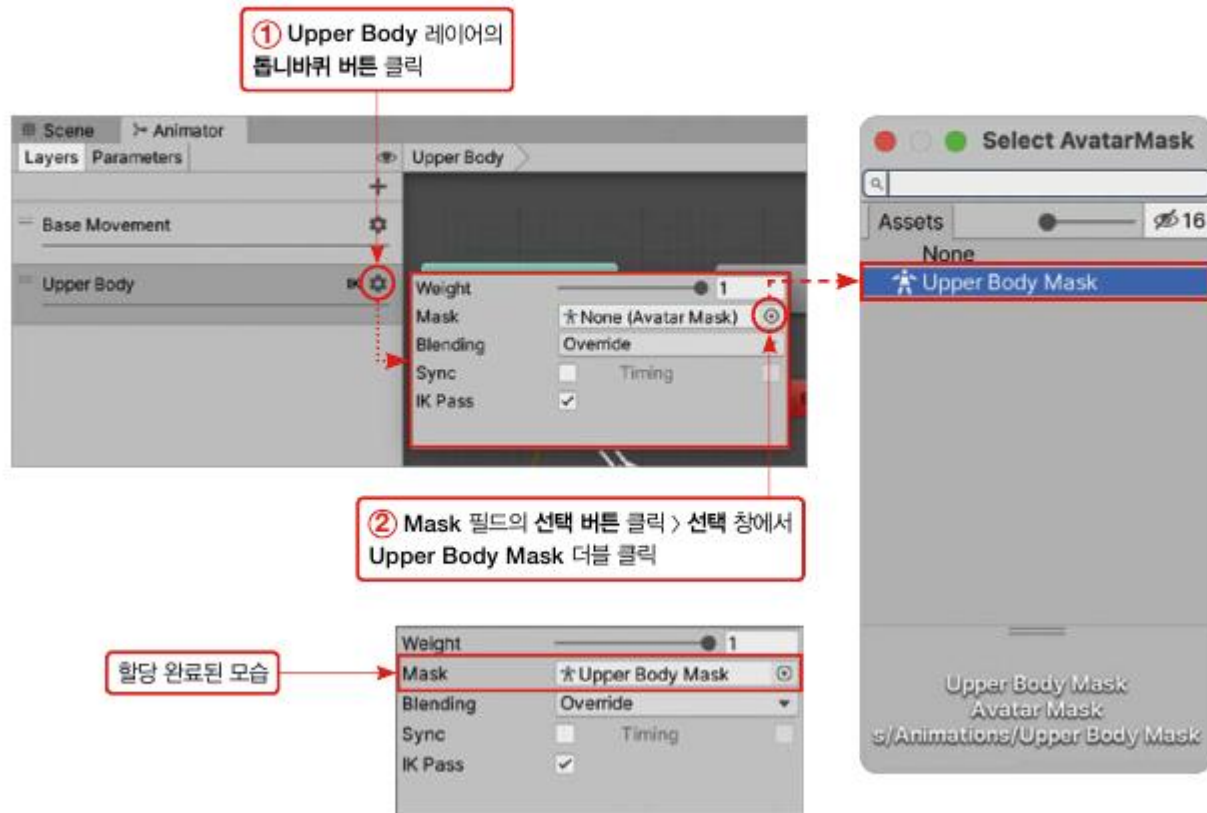
이 접근법을 역운동학(IK)이라고 하며, 올바르게 설정된 아바타가 있는 휴머노이드 캐릭터의 메카닉에서 지원.



플레이어 캐릭터와 애니메이션

[과정 1] Upper Body 레이어에 아바타 마스크 적용

- ① Upper Body 레이어의 톱니바퀴 버튼 클릭
- ② Mask 필드의 선택 버튼 클릭 > 선택 창에서 Upper Body Mask 더블 클릭

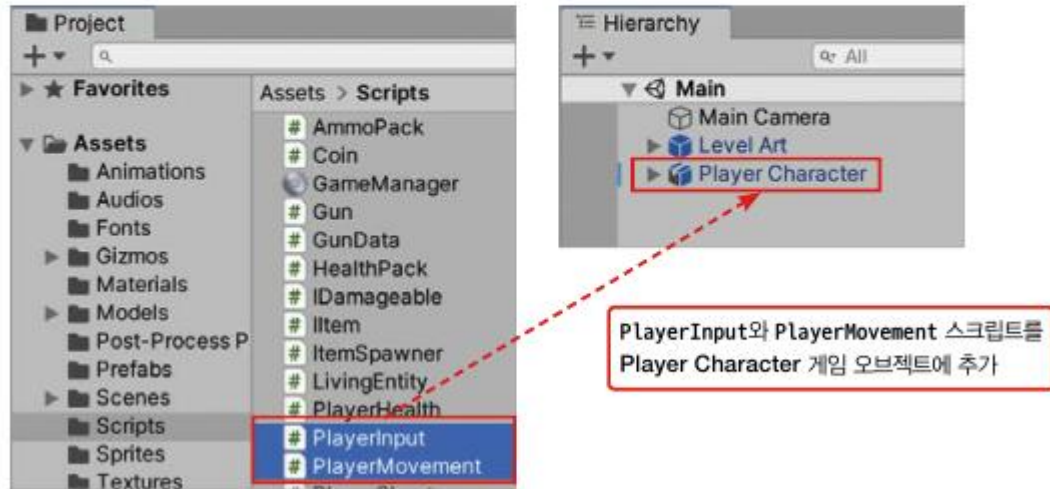


▶ Upper Body 레이어에 아바타 마스크 적용하기

플레이어 캐릭터와 애니메이션

[과정 I] 캐릭터 이동을 위한 스크립트 추가

- ① 프로젝트 창에서 Scripts 폴더로 이동
- ② PlayerInput 스크립트를 하이어라키 창의 Player Character로 드래그&드롭
- ③ PlayerMovement 스크립트를 하이어라키 창의 Player Character로 드래그&드롭



▶ 캐릭터 이동을 위한 스크립트 추가

플레이어 캐릭터와 애니메이션

▼ Fire1	
Name	Fire1
Descriptive Name	
Descriptive Negative	
Negative Button	
Positive Button	left ctrl
Alt Negative Button	
Alt Positive Button	mouse 0
Gravity	1000
Dead	0.001
Sensitivity	1000
Snap	<input type="checkbox"/>
Invert	<input type="checkbox"/>

▼ Reload	
Name	Reload
Descriptive Name	
Descriptive Negative	
Negative Button	
Positive Button	r
Alt Negative Button	
Alt Positive Button	
Gravity	1000
Dead	0.001
Sensitivity	1000

▶ 입력 매니저에서 확인한 Fire1과 Reload

플레이어 캐릭터와 애니메이션

[과정 1] PlayerMovement의 Start() 메서드 완성하기

① PlayerMovement 스크립트의 Start() 메서드를 다음과 같이 완성

```
private void Start() {  
    // 사용할 컴포넌트의 참조 가져오기  
    playerInput = GetComponent<PlayerInput>();  
    playerRigidbody = GetComponent<Rigidbody>();  
    playerAnimator = GetComponent<Animator>();  
}
```

[과정 2] PlayerMovement의 FixedUpdate() 메서드 완성하기

① PlayerMovement 스크립트의 FixedUpdate() 메서드를 다음과 같이 완성

```
private void FixedUpdate() {  
    // 회전 실행  
    Rotate();  
    // 움직임 실행  
    Move();  
  
    // 입력값에 따라 애니메이터의 Move 파라미터값 변경  
    playerAnimator.SetFloat("Move", playerInput.move);  
}
```

[과정 3] PlayerMovement의 Move() 메서드 완성하기

① PlayerMovement 스크립트의 Move() 메서드를 다음과 같이 완성

```
private void Move() {  
    // 상대적으로 이동할 거리 계산  
    Vector3 moveDistance =  
        playerInput.move * transform.forward * moveSpeed * Time.deltaTime;  
    // 리지드바디를 이용해 게임 오브젝트 위치 변경  
    playerRigidbody.MovePosition(playerRigidbody.position + moveDistance);  
}
```

[과정 4] PlayerMovement의 Rotate() 메서드 완성하기

① PlayerMovement 스크립트의 Rotate() 메서드를 다음과 같이 완성

```
private void Rotate() {  
    // 상대적으로 회전할 수치 계산  
    float turn = playerInput.rotate * rotateSpeed * Time.deltaTime;  
    // 리지드바디를 이용해 게임 오브젝트 회전 변경  
    playerRigidbody.rotation =  
        playerRigidbody.rotation * Quaternion.Euler(0, turn, 0f);  
}
```

플레이어 추적 카메라

플레이어 추적 카메라

- 시네머신

com.unity.cinemachine

시네머신은 Unity 카메라 작동을 위한 모듈 제품군으로 타겟 추적, 구성, 블렌딩, 샷 간 전환의 복잡한 수학 및 논리 문제를 해결. 개발 중에 발생하는 시간이 많이 걸리는 수동 조작과 스크립트 수정 횟수를 크게 줄이도록 설계.

씬에서 애니메이션, 차량 속도, 터레인 또는 기타 게임 오브젝트를 변경하는 등의 조정을 수행하면 시네머신은 해당 동작을 동적으로 조정. (예: 캐릭터가 오른쪽이 아니라 왼쪽으로 돈다는 이유로 카메라 스크립트를 다시 작성할 필요가 없음.)

시네머신은 FPS, 3인칭, 2D, 횡스크롤 게임, 톱다운 뷰, RTS를 포함한 모든 장르에서 실시간으로 동작하고, 씬에서 필요한 만큼의 샷을 지원. 이러한 모듈식 시스템을 이용하여 정교한 동작을 구성.

시네머신은 다른 Unity 툴과 잘 연동됨. 타임라인, 애니메이션 및 포스트 프로세싱 에셋에 대한 강력한 보완재 역할. 고유한 확장을 생성하거나 커스텀 카메라 스크립트와 통합.

플레이어 추적 카메라



▶ 시네머신과 타임라인 에디터를 함께 사용한 모습

플레이어 추적 카메라

- **시네머신 브레인(브레인 카메라)**

게임 월드를 촬영하는 '진짜' 카메라이며 씬에 하나만 존재

Unity 카메라의 컴포넌트

씬에 있는 액티브 가상 카메라를 모두 모니터링

원하는 가상 카메라의 게임 오브젝트를 활성화 또는 비활성화. 시네머신 브레인이 라이브 가상 카메라와 같거나 더 높은 우선 순위를 지닌 가장 최근에 활성화된 가상 카메라를 선택(한번에 하나의 가상 카메라만 현재 활성화된 카메라로 사용 가능)

이전 가상 카메라와 새 가상 카메라 간의 컷이나 블렌딩을 수행
(블렌딩하는 동안에는 두 가상 카메라가 모두 라이브 상태)

플레이어 추적 카메라

- 가상 카메라

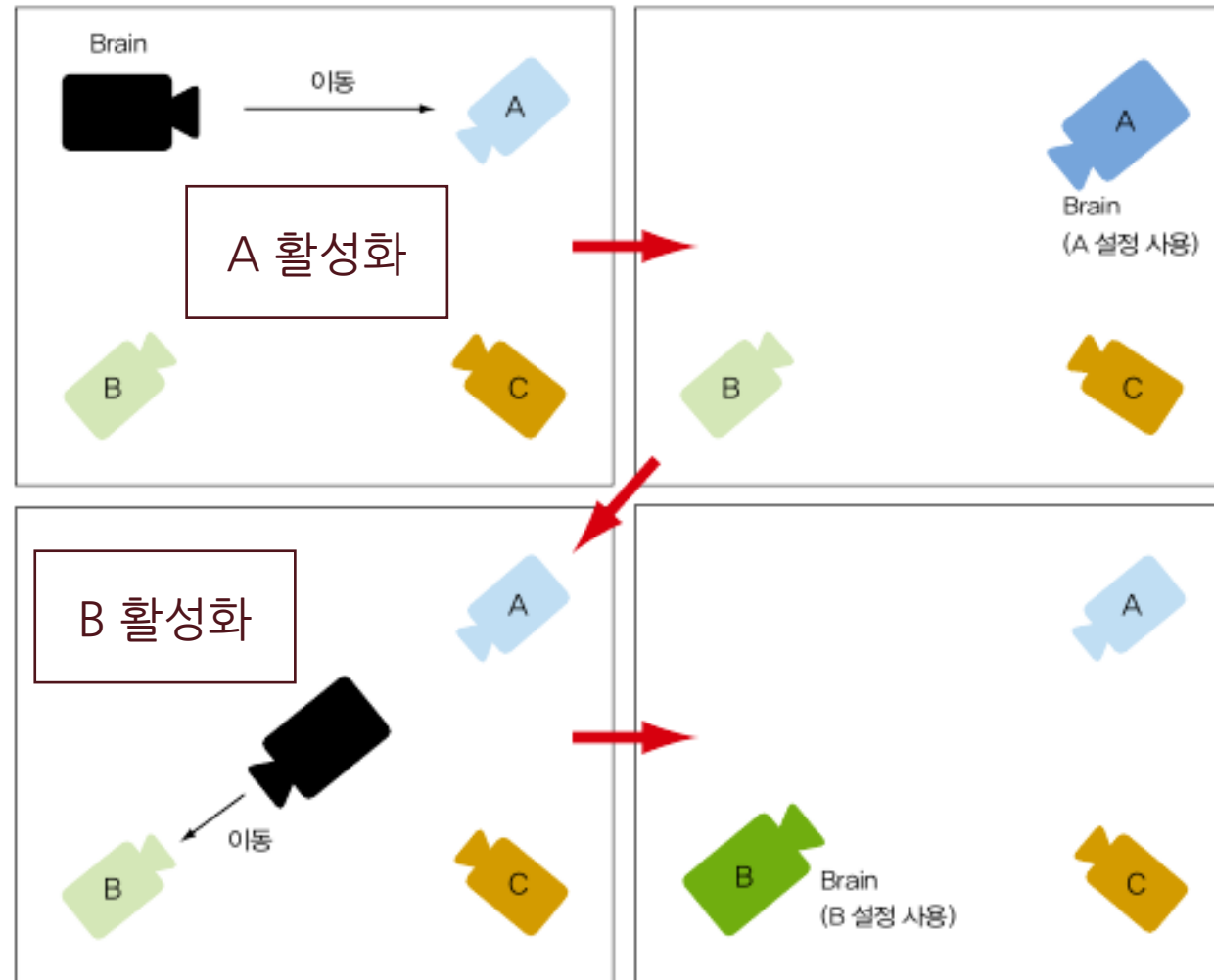
가상 카메라는 Unity 카메라(브레인 카메라)를 이동 및 회전하고 해당 설정을 제어

가상 카메라는 Unity 카메라와 별개의 게임 오브젝트이며, 독립적으로 동작

가상 카메라로 수행할 수 있는 주요 작업

1. 씬에 Unity 카메라를 배치.
2. 무언가를 향해 Unity 카메라를 조준.
3. Unity 카메라에 절차적 노이즈를 추가.
노이즈는 핸드헬드 효과 또는 차량 흔들림 등을 시뮬레이션.

플레이어 추적 카메라



▶ 시네머신 카메라의 동작 원리

플레이어 추적 카메라

- 이동 및 조준

- 타겟

- Follow - 가상 카메라가 함께 따라 이동할 게임 오브젝트를 지정합니다.

- Look At - 게임 오브젝트가 조준할 게임 오브젝트를 지정합니다.

- Body 프로퍼티

- 씬에서 움직이는 방식을 지정(카메라가 Follow에 할당된 추적대상을 어떻게 따라다닐지 결정)

- Aim 프로퍼티

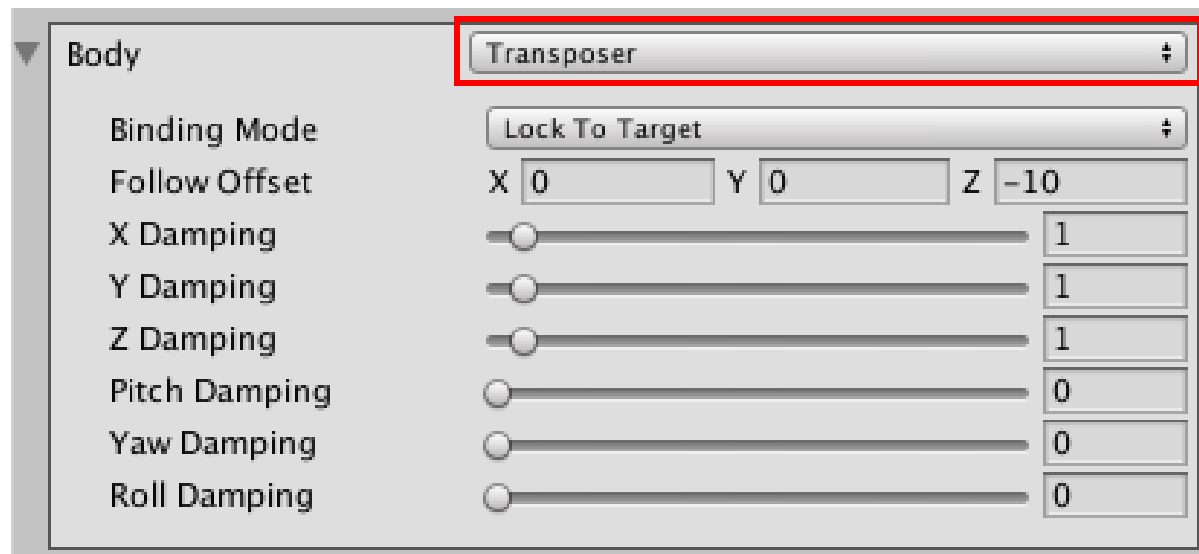
- 회전 방식을 지정(카메라가 조준하는 지점을 설정)

- Noise 프로퍼티

- 가상 카메라의 움직임에 노이즈를 추가

플레이어 추적 카메라

- Body 프로퍼티

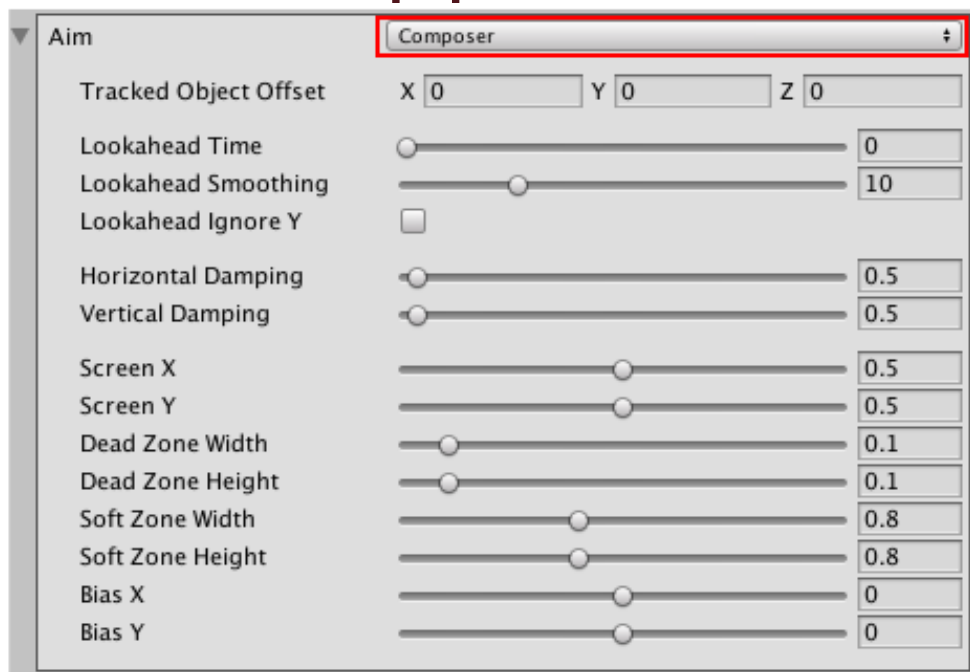


시네머신에는 가상 카메라를 움직이는 다음의 알고리즘이 포함되어 있습니다.

- **Transposer: Follow** 타겟과 고정된 관계로 움직입니다.
- **Do Nothing:** 가상 카메라를 움직이지 않습니다.
- **Framing Transposer: Follow** 타겟과 고정된 스크린 공간 관계로 움직입니다.
- **Orbital Transposer: Follow** 타겟과 가변 관계로 움직이며, 선택적으로 플레이어 입력을 허용합니다.
- **Tracked Dolly:** 사전 정의된 경로를 따라 움직입니다.
- **Hard Lock to Target: Follow** 타겟과 동일한 포지션을 사용합니다.

플레이어 추적 카메라

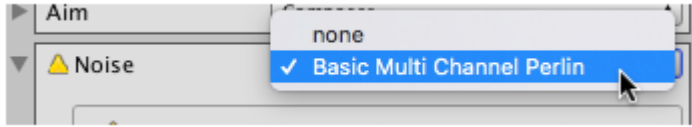
- Aim 프로퍼티



- **Composer:** 카메라 프레임에 **Look At** 타겟을 유지합니다.
- **Group Composer:** 카메라 프레임에 여러 개의 **Look At** 타겟을 유지합니다.
- **Do Nothing:** 가상 카메라를 절차적으로 회전하지 않습니다.
- **POV:** 사용자의 입력에 따라 가상 카메라를 회전합니다.
- **Same As Follow Target:** 카메라의 회전을 **Follow** 타겟의 회전으로 설정합니다.
- **Hard Look At:** 카메라 프레임의 중간에 **Look At** 타겟을 유지합니다.

플레이어 추적 카메라

- Noise 프로퍼티



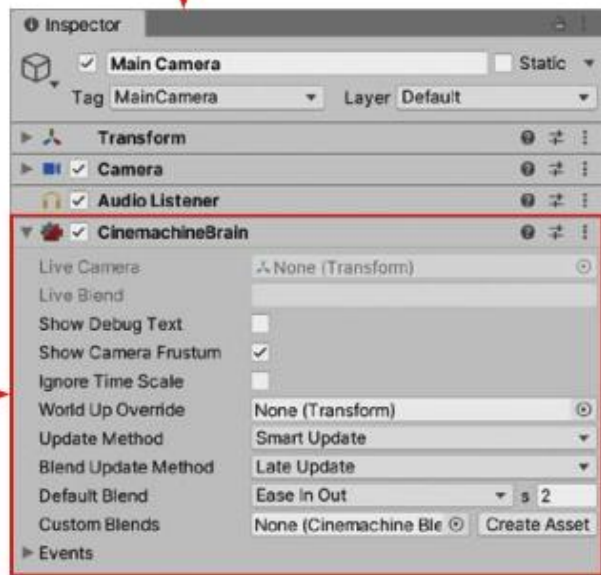
1. 씬 뷰 또는 계층 창에서 가상 카메라를 선택합니다.
2. 인스펙터에서 **Noise** 드롭다운 메뉴를 사용하여 **Basic Multi Channel Perlin** 을 선택합니다.
3. **Noise Profile** 에서 기존 프로파일 에셋을 선택하거나 **고유한 프로파일을 생성**합니다.
4. **Amplitude Gain** 과 **Frequency Gain** 을 사용하여 노이즈를 미세 조정합니다.

플레이어 추적 카메라

[과정 01] 브레인 카메라와 가상 카메라 만들기

- ① 하이어라키 창에서 Main Camera 게임 오브젝트 선택
- ② Cinemachine Brain 컴포넌트 추가(Add Component > Cinemachine > Cinemachine Brain)
- ③ 새 가상 카메라 생성(하이어라키 창에서 + > Cinemachine > Virtual Camera 클릭)
- ④ 생성된 가상 카메라 게임 오브젝트의 이름을 Follow Cam으로 변경

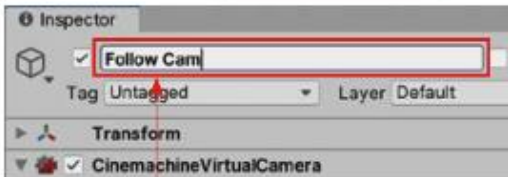
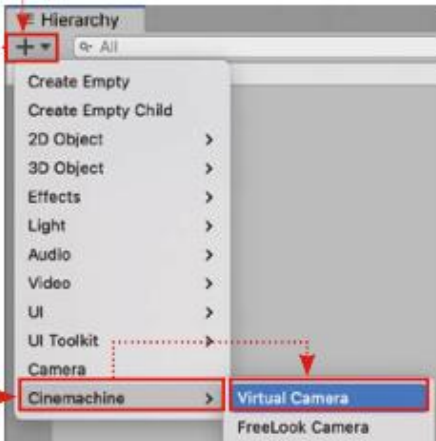
① Main Camera 게임 오브젝트 선택



② Cinemachine Brain 컴포넌트 추가(Add Component > Cinemachine > Cinemachine Brain)

▶ 브레인 카메라와 가상 카메라 만들기

③ 새 가상 카메라 생성(하이어라키 창에서 + > Cinemachine > Virtual Camera 클릭)

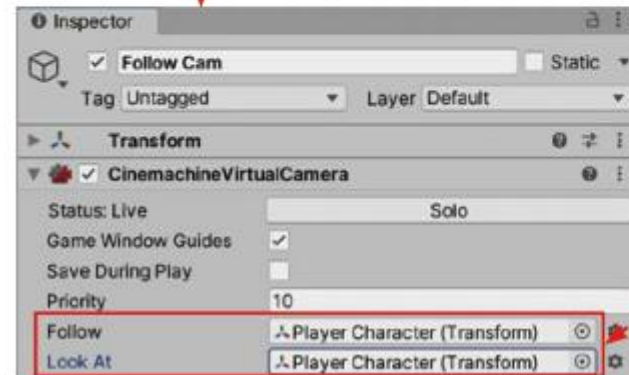


④ 생성된 가상 카메라 게임 오브젝트의 이름을 Follow Cam으로 변경

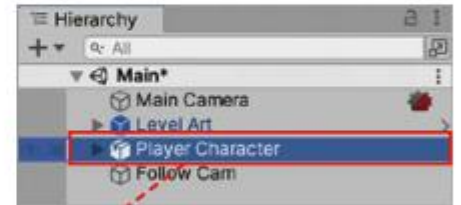
[과정 02] Follow Cam의 추적 대상 설정하기

- ① 하이어라키 창에서 Follow Cam 게임 오브젝트 선택
- ② Cinemachine Virtual Camera 컴포넌트의 Follow 필드와 Look At 필드에 Player Character 게임 오브젝트 드래그&드롭

① Follow Cam 게임 오브젝트 선택



▶ Follow Cam의 추적 대상 설정하기



② Follow 필드와 Look At 필드에 Player Character 게임 오브젝트 드래그&드롭

플레이어 추적 카메라

DeadZone

카메라가 타겟의 움직임을 무시하는 화면 영역
카메라가 회전하지 않음

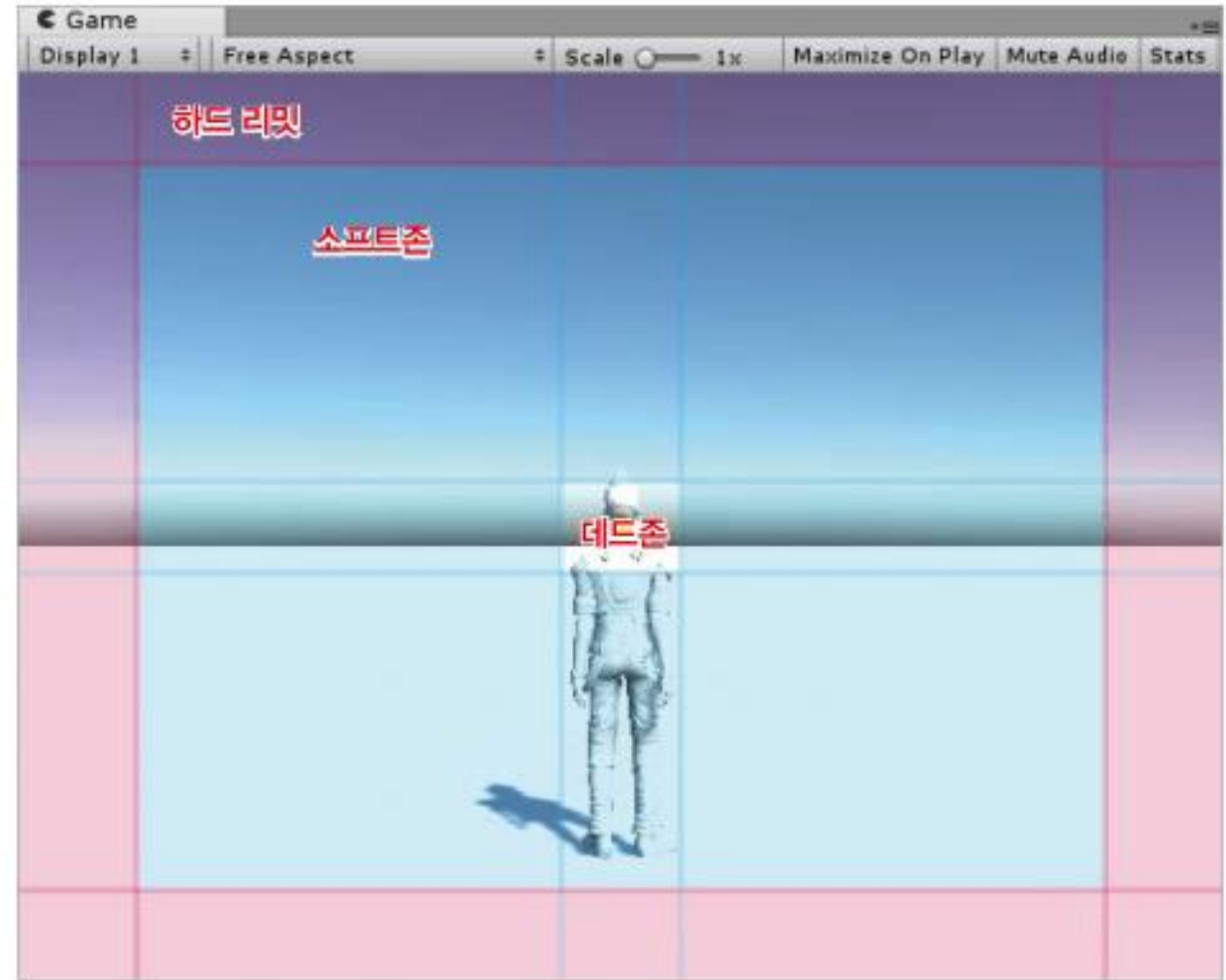
SoftZone

물체가 화면의 조준점에 오도록 카메라가 부드럽게 회전
Horizontal Damping, Vertical Damping 설정시간에
맞춰 카메라를 데드존으로 밀어냄

HardLimit

물체가 너무 빠르게 움직여 화면의 SoftZone을 벗어나
HardLimit에 도달하려 한다면 카메라가 격하게 회전하
여 SoftZone을 벗어나지 않도록 함

데드존과 소프트존의 크기를 줄이면 물체가 조금이라도
화면 중앙을 벗어나려 할 때 지연시간 없이 카메라가 즉
시 물체를 향해 회전

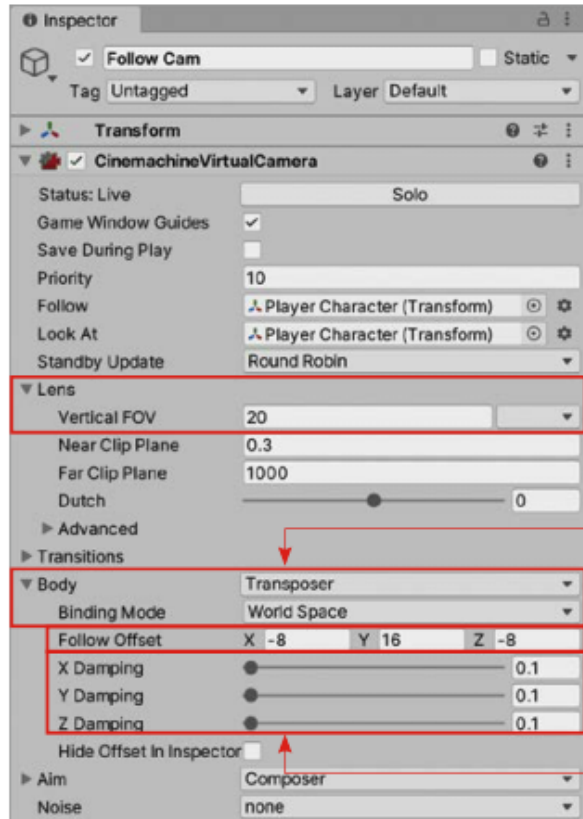


▶ 데드존, 소프트존, 하드 리밋

플레이어 추적 카메라

[과정 이] 가상 카메라 시야각과 Body 파라미터 설정하기

- ① Field Of View를 20으로 변경
- ② Body 탭 펼치기 > Binding Mode를 World Space로 변경
- ③ Follow Offset을 (-8, 16, -8)로 변경
- ④ X Damping, Y Damping, Z Damping을 0.1로 변경



▶ 가상 카메라 Body 파라미터 설정하기



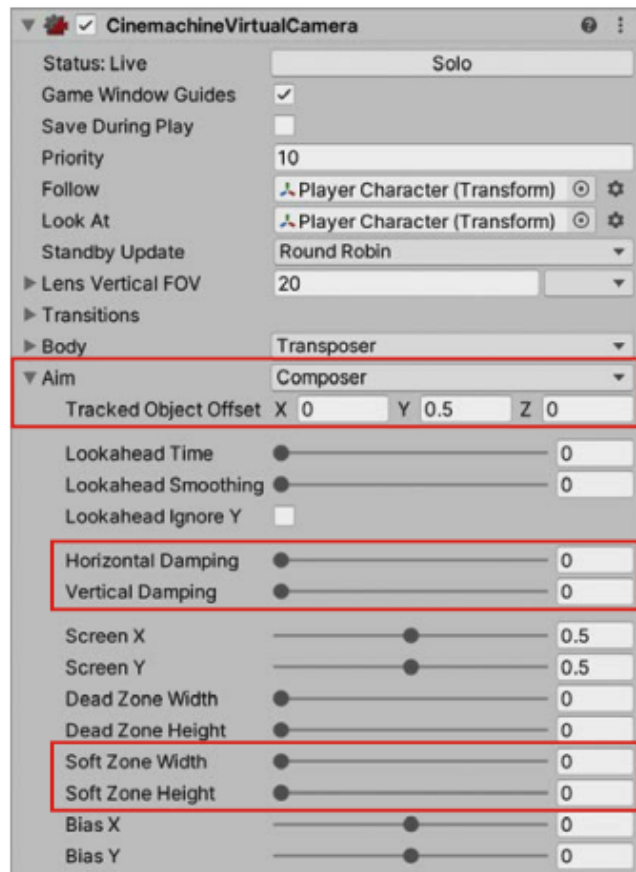
▶ 시야각



플레이어 추적 카메라

[과정 02] 가상 카메라의 Aim 파라미터 설정하기

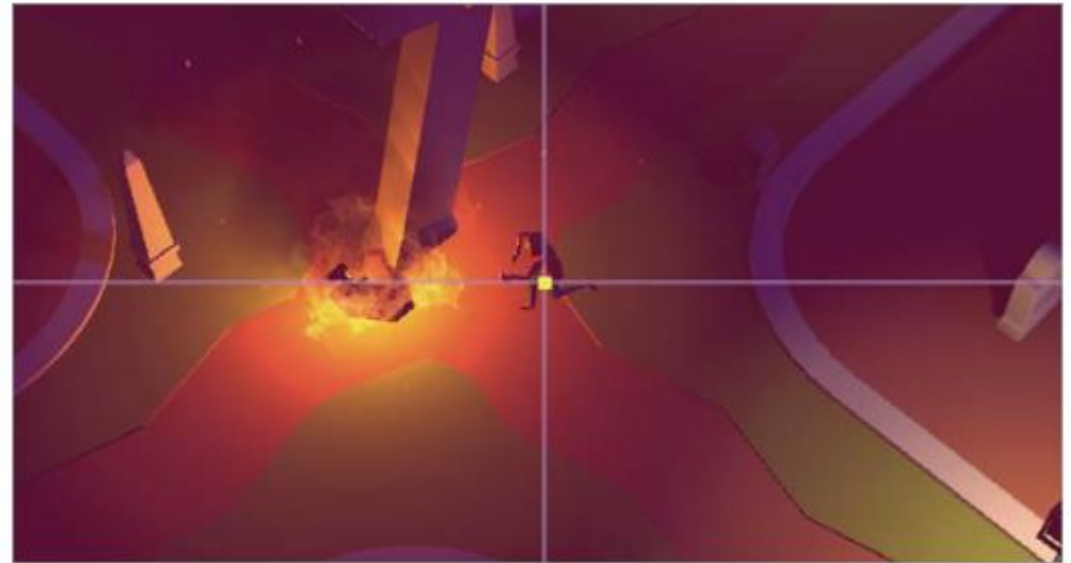
- ① Tracked Object Offset을 (0, 0.5, 0)으로 변경
- ② Horizontal Damping과 Vertical Damping을 0으로 변경
- ③ Soft Zone Width와 Soft Zone Height를 0으로 변경



① Aim 탭 펼치기 > Tracked Object Offset을 (0, 0.5, 0)으로 변경

② Horizontal Damping과 Vertical Damping을 0으로 변경

③ Soft Zone Width와 Soft Zone Height를 0으로 변경



▶ 추적 카메라 완성

▶ 가상 카메라의 Aim 파라미터 설정하기

RiderFlow

RiderFlow

1. 모든 위치에서 모든 항목을 검색

Unity 에디터에서 씬의 게임 객체, 파일, 액션 등을 프로젝트 전체에서 신속하게 검색.

2. 씬 구조의 시각적 이해

씬, 다른 애셋 및 프리팹에서 애셋이 사용되는 위치를 확인하려면 Find Usages(사용 위치 찾기)를 사용.

3. 복잡한 씬 처리

계층 구조 패널에서 객체를 색상별로 그룹화하여 복잡한 씬을 한층 쉽게 처리.

북마크를 추가하면 나중에 다시 볼 수 있음.

4. 효과적인 씬 관리

씬 뷰 관리 툴바는 복잡한 씬 탐색을 위한 나침반과 같은 역할.

씬 활동을 효율적으로 관리할 수 있도록 다양한 도구가 제공.

씬에서 사용 가능한 애셋만 표시하고 검색 결과의 애셋을 씬으로 바로 드래그 가능한 검색 도구를 활용.

현재 카메라 위치를 프리셋으로 저장, 단축키를 할당, 프리셋 간 전환하는 데 도움이 되는 카메라 액션 도구도 지원.

5. 안정적인 리팩터링

Replace objects(객체 바꾸기) 리팩터링은 씬에서 한 개 이상의 객체를 빠르게 바꾸고 참조를 자동으로 업데이트

6. 스마트 에디터의 다양한 기능 활용

RiderFlow는 C# 스크립트 파일의 사소하고 간단한 변경을 위해 설계된 간단한 버전의 에디터를 제공.

(Rider for Unity와 충돌)