Unity®

UNITY 2022 LTS 에디션 → 전자책

Unity로 제작한 캐릭터, Sakura Rabbit

Unity 애니메이션 집중 탐구 가이드

목차

들어가는 말

Unity 단편 애니메이션 에너미즈의 스크린샷

애니메이션은 게임 디자인에서 매우 중요한 부분이며, 현재 많은 게임이 영화에 가까운 비주얼 품질의 애니메이션 콘텐츠를 선보이고 있습니다.

애니메이터는 캐릭터 묘사와 비주얼 스토리텔링을 통해 게임의 스토리를 전달하며, 이를 위해 캐릭터 리깅과 포즈 작업을 진행하고 애니메이션 시퀀스의 키프레임을 설정합니다. 게임 개발에서 애니메이터의 역할이 진화함에 따라 Unity의 애니메이션 툴과 워크플로도 함께 발전하고 있습니다.

이 가이드는 Unity 애니메이션 시스템의 전용 애니메이션 툴, 메서드, 기법을 사용하는 방법을 알고자 하는 숙련된 애니메이터와 초보 애니메이터 모두를 위해 작성되었습니다.

Unity로 애니메이션을 임포트하는 방법, 일반 및 휴머노이드 애니메이션 유형, 에디터에서 애니메이션을 만들고 리깅하는 방법, 타임라인으로 애니메이션 컷씬을 제작하는 방법, 고급 애니메이션 옵션 사용 방법 등을 이 가이드에서 배울 수 있습니다.

사실적인 씬, 핸드 페인팅 및 셀 셰이딩, 카툰 비주얼 등 스타일에 상관없이 Unity에서 애니메이션을 구현할 수 있습니다.

런타임 리깅으로 환경에 반응하는 애니메이션화된 캐릭터

도움을 주신 분들

피트 젭슨은 지난 10년 동안 영화 제작, 애니메이션, 사진, 그래픽 디자인, 게임 디자인과 같은 크리에이티브 미디어 과목을 가르친 전문 강사입니다. 10년 동안의 Unity 경험을 바탕으로 Udemy.com에서 여러 디자인 강좌를 제공하고 있습니다.

도움을 주신 유니티 직원 분들

데이브 헌트는 유니티 코펜하겐에서 애니메이션과 리깅을 담당하고 있는 테크니컬 아티스트입니다. 데이브는 재사용 가능한 파이프라인과 원활한 상호 호환성을 바탕으로 아티스트와 애니메이터를 지원하는 데 집중하고 있습니다. 시애틀에서 18년 동안 게임 개발에 종사했으며 2017년에 유니티에 입사했습니다. Bungie의 리깅 테크 아트 팀 리더로서 헤일로와 데스티니 시리즈 게임 제작에 기여했고, 워싱턴 대학교의 애니메이션 캡스톤 프로그램에서 시간 강사로 일하기도 했습니다.

데이브 헌트의 머리글

애니메이션은 창작을 통한 표현 방식 중에서도 무척 진실된 형태가 아닐까 생각합니다. 제대로 구현된 애니메이션은 현실처럼 느껴지는 착각을 불러옵니다. 이제 고전이 된 저서 Illusion of Life: Disney Animation에서 밝힌 대로 프랭크 토마스는 디즈니 초창기 시절 올리 존스턴 및 다른 동료들과 함께 애니메이션의 기본 원칙을 개척 및 문서화했고, 이러한 원칙은 현재 전문 애니메이터들도 사용하고 있습니다.

어릴 때 저는 점토와 나무로 만든 장난감 로봇과 피규어로 스톱 모션 영상을 제작하면서 애니메이션의 즐거움을 알게 되었습니다. 아이디어를 실현하는 작업이 즐거웠던 저는 더 큰 규모의 프로젝트에 도전했고, 이를 계기로 워싱턴 대학교의 컴퓨터 애니메이션 캡스톤 프로그램에 참여하게 되었습니다. 당시의 결정으로 인해 프랭크 토마스가 캘리포니아 버뱅크에 있는 본인의 자택에서 직접 제 학창 시절 애니메이션 프로젝트를 검토해 주는 기회를 가지게 될 줄은 정말 몰랐습니다. 당시 제 프로젝트를 검토하며 애니메이션으로 제작한 개를 실제 개처럼 보이게 만드는 것이 무엇인지 어린아이처럼 반짝이던 눈으로 설명하던 91세의 프랭크 토마스를 결코 잊지 못할 것입니다. 그는 끊임없이 생명체를 관찰하고 그 움직임에 어떤 섬세한 아름다움이 있는지를 기록하여 애니메이션 작업에 활용했습니다.

진정한 대가를 마주한 이래로 저는 애니메이션의 즐거움을 널리 알리고 싶어졌습니다. 운이 좋게도 저는 15년 넘게 워싱턴 대학교의 캡스톤 프로그램에서 애니메이션을 가르치게 되었고, 2000년부터는 정식으로 게임 개발 업계의 애니메이터이자 테크니컬 아티스트로 일할 수 있었습니다. 그리고 저는 AAA 게임 스튜디오의 직원이든, 친구들과 소규모 팀을 결성해 인디 게임 잼에 참여하는 사람이든 애니메이션 제작에서 느끼는 즐거움은 같을 것이라 생각합니다.

이 전자책의 저자들과 협업함으로써 초심자와 전문가에게 인터랙티브 애니메이션을 통한 멋진 창작의 기회를 선사하는 Unity 애니메이션을 소개할 수 있게 되어서 무척 기쁜 마음입니다.

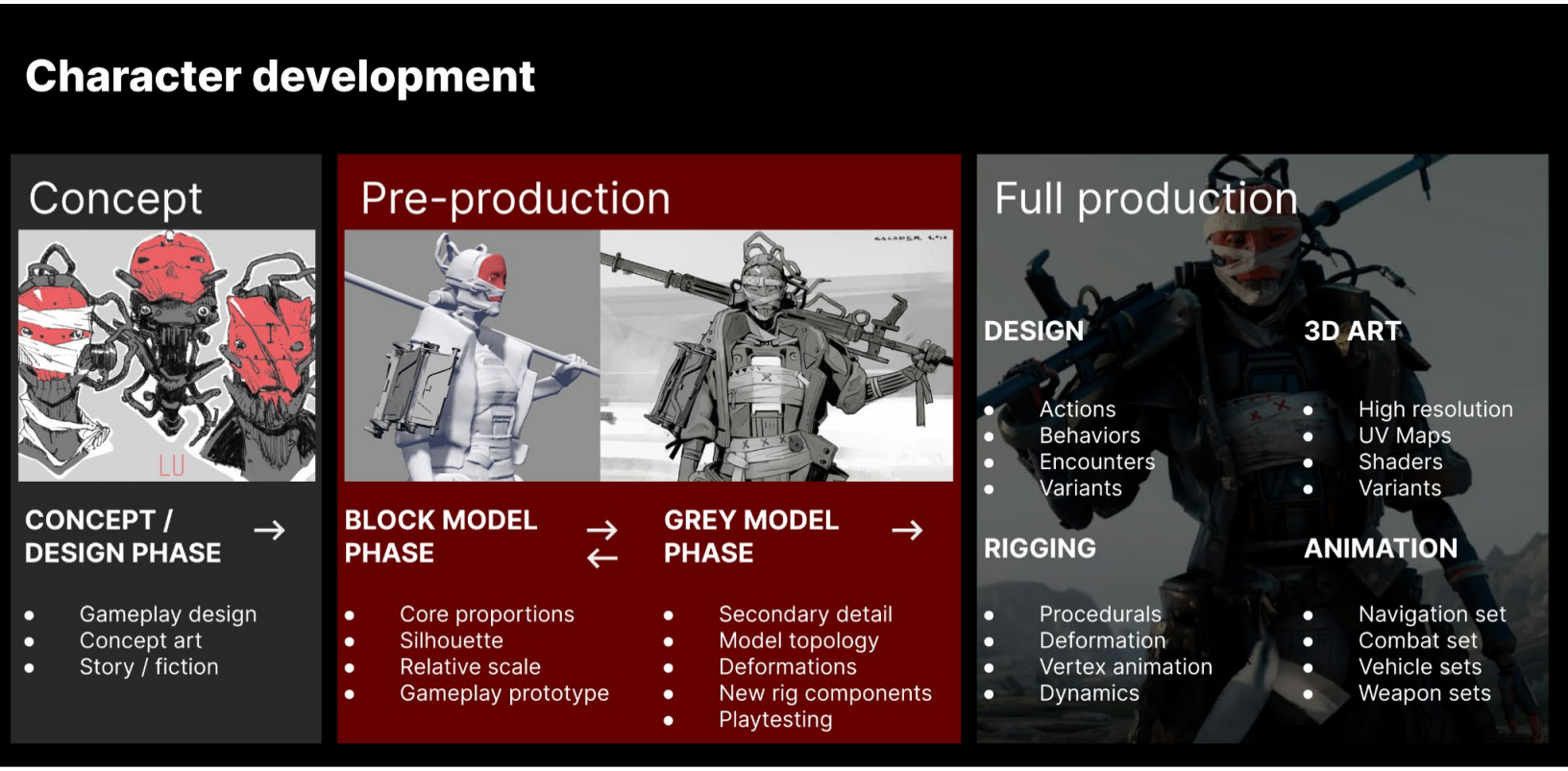
절친한 친구 제이와 함께 개구리 캐릭터 럽츠가 등장하는 Unity 3용 3인칭 플랫포머 튜토리얼을 처음 다운로드했을 때가 기억납니다. 당시 한 서적의 어떤 챕터에는 걷기 애니메이션 클립을 재생하면서 플레이어 입력을 받아 캐릭터를 앞으로 움직이게 해야 하는 몇 줄의 코드가 수록되어 있었습니다. 해당 코드를 사용하여 저희가 상상했던 스토리, 월드, 인터랙티브 경험을 구현할 수 있었죠. 이미 베테랑 애니메이터이자 게임 개발자였음에도, 그 짧은 설명이 새로운 컨텍스트에서 작업을 시작하는 데 도움이 되었던 것입니다. 이 전자책을 통해 여러분도 Unity 애니메이션 툴을 다루는 방법을 익히게 되기를 바라며, 여러분이 구현하게 될 작품을 기대하겠습니다.

성공을 꿈꾸는 게임을 위한 애니메이션 제작

많은 게임에는 다양한 캐릭터, 반전과 전환이 있는 스토리라인, 다면적인 환경이 포함되어 있으며 이 모든 요소는 마찬가지로 다양한 개발자 및 아티스트로 구성된 팀에서 제작되는 경우가 많습니다. 콘텐츠 크리에이터를 위해 프로젝트에 적합한 툴을 갖춘 견고한 애니메이션 파이프라인을 구축하면 수천 개의 애니메이션을 관리할 수 있어 대규모 프로젝트에 도움이 됩니다.

캐릭터 컨셉을 구상한 다음 모델링, 리깅, 애니메이션 제작으로 넘어가는 프로세스는 게임 개발에서 캐릭터를 제작하기 위해 널리 활용되는 절차입니다. 하지만 이 절차는 완벽하지 않으며 일반적으로 예술 작품을 제작하는 과정은 그렇게 간단하지 않습니다.

예를 들어 캐릭터 컨셉의 다리가 너무 길어 자연스러운 이동(locomotion) 사이클을 구현하지 못할 수도 있고, 팔과 몸통을 잘못 조합하면 캐릭터가 무기를 사실적인 형태로 잡지 못할 수도 있습니다. 이상적인 캐릭터 제작 프로세스라면 캐릭터 디자인에서 비파괴적인 반복 작업(iteration)이 가능해야 합니다. 데이브 헌트와 포레스트 쇠더린드의 GDC 2015 발표에서 이들이 작업 과정을 어떻게 구축했는지 살펴볼 수 있습니다. 이들은 대규모 제작에 적합한 시스템을 개발했습니다. 이 시스템은 게임플레이, 스토리텔링, 월드 구축의 효율적인 결합이 가능한 Maya의 절차적 제어 모듈식 릭(rig) 제작 프레임워크를 사용합니다. 아래 이미지에서 작업 과정의 주요 단계를 확인할 수 있습니다. 이 워크플로는 Unity에서도 적용할 수 있습니다.



GDC 2015 연설에서 소개한 캐릭터 개발의 주요 단계, 캐릭터 이미지 출처: Unity 데모 아담

애니메이터 지원

게임 개발에서는 테크니컬 아티스트가 아티스트, 개발자, 디자이너 간의 효율적인 협업을 지원하는 경우가 많습니다.

스크립팅, 리깅 및 테크니컬 아트 분야에서는 대규모 제작 시 자동화된 프로세스를 통해 수천 개의 파일을 관리할 수 있습니다.

다음은 애니메이션 파일을 정리하기 위한 몇 가지 일반적인 팁입니다. 하지만 모든 프로젝트는 근본적으로 서로 다르기 때문에 항상 여러분의 작업에 적합한 파이프라인을 만들어야 합니다.

— **명명 규칙**: 캐릭터는 많은 수의 오브젝트, 지오메트리, 뼈대, 액세서리로 구성됩니다. 모든 팀원이 계층 구조를 쉽게 탐색할 수 있도록 표준화된 이름을 사용하는 것이 권장됩니다. 단순하면서도 쉽게 알아볼 수 있는 명명 규칙을 수립하세요. 팀의 애니메이터를 위한 커스텀 툴을 제작한 경우에도 표준에 따라 쉽게 파악할 수 있는 이름을 지정하는 것이 유용할 것입니다.

— **씬 구성**: Unity에서 직접 작업하려는 애니메이터라면 샌드박스 씬을 만들거나 공통적인 게임플레이 애니메이션을 프리팹 모드에서 작업하는 것을 고려해 보세요.

— **에셋 버전 추적 및 자동화:** Unity의 AssetPostprocessor 클래스를 사용하면 에셋 임포트 시 코드를 실행하거나 프리셋을 사용하여 자동 임포트 설정을 적용할 수 있습니다. 그러면 에셋이 팀의 표준을 준수하는지 효율적으로 검증할 수 있으므로 실제 콘텐츠 제작에 더 집중할 수 있습니다.

— **Unity에서 초안 제작 및 FBX 익스포터 사용:** 디자이너는 Unity에서 대략적인 의도와 타이밍을 효율적으로 표현할 수 있는 타임라인 같은 시스템을 사용하여 애니메이션과 시네마틱 시퀀스를 모의로 만들어 볼 수 있습니다. 그렇게 제작된 프로토타입 애니메이션은 FBX 익스포터를 통해 애니메이터가 선호하는 DCC 소프트웨어로 익스포트하여 추가로 다듬을 수 있습니다.

— **시각화 및 커스텀 에디터 툴**: Unity는 에디터 툴을 통해 뛰어난 유연성을 제공합니다. 필요에 따라 비주얼 컨트롤 릭 같은 커스텀 인터페이스나 Unity에서 아티스트가 다양한 애니메이션 툴의 API를 간편하게 사용할 수 있는 툴을 제작할 수 있습니다.

— **Animation Rigging 패키지를 사용한 Unity의 IK:** 런타임 리깅을 활성화하면 캐릭터가 게임 월드와 연결됩니다. 손으로 표면을 쓸거나 문손잡이를 잡고 돌리는 것은 뼈대 체인을 수정하여 사실적으로 표현할 수 있는 섬세한 동작의 예시 중 하나입니다. Unity의 Animation Rigging 패키지를 사용하면 이처럼 세밀한 움직임을 제작할 수 있어 캐릭터 제작에 큰 도움이 됩니다.

Unity의 애니메이션 시스템

Unity를 처음 사용하는 경우 Unity Learn 튜토리얼을 통해 에디터를 탐색하고 사용하는 방법을 배울 수 있으며, 특히 3개의 학습 길잡이 교육 과정인 Unity 필수 과정, 주니어 프로그래머, 크리에이티브 코어가 유용합니다.

Unity 매뉴얼도 애니메이션에 관하여 종합적으로 정리한 섹션이 제공되는 중요한 리소스입니다. 다음은 매뉴얼에서 발췌한 간략한 소개입니다.

Unity의 애니메이션 시스템에서 제공하는 기능은 다음과 같습니다.

— 오브젝트, 캐릭터, 프로퍼티 등 Unity의 모든 요소에 대한 간편한 애니메이션 워크플로 및 설정

— 임포트한 애니메이션 클립 및 Unity에서 제작한 애니메이션에 대한 지원

— 휴머노이드 애니메이션 리타게팅 - 한 캐릭터 모델의 애니메이션을 다른 캐릭터 모델에 적용하는 기능

— 애니메이션 클립 정렬을 위한 간소화된 워크플로

— 애니메이션 클립, 전환 및 상호 작용의 간편한 미리 보기

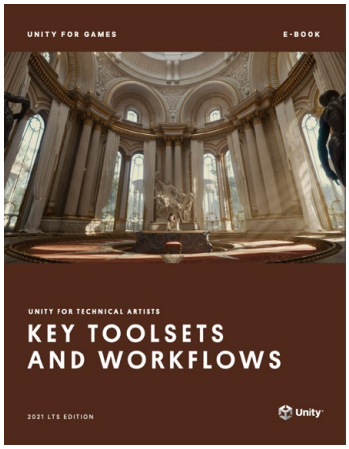
— 비주얼 프로그래밍 툴을 활용한 애니메이션 간의 복잡한 상호 작용 관리

— 다양한 로직을 통해 서로 다른 신체 부위에 애니메이션을 적용하는 기능

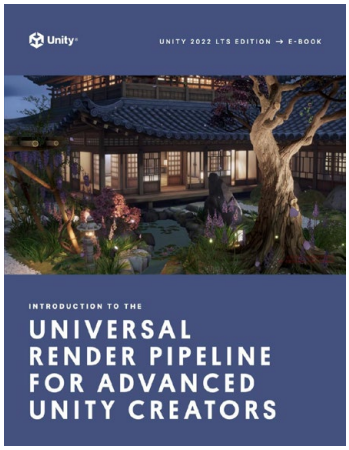
— 레이어링 및 마스킹 기능

이 가이드의 스크린샷은 SRP(스크립터블 렌더 파이프라인) 프레임워크를 기반으로 제작된 Unity의 멀티플랫폼 렌더링 솔루션인 URP(유니버설 렌더 파이프라인)로 제작되었습니다. URP는 Unity에서 사용할 수 있는 3가지 렌더 파이프라인 중 하나이며, 다른 2가지는 각각 HDRP(고해상도 렌더 파이프라인) 및 빌트인 렌더 파이프라인입니다.

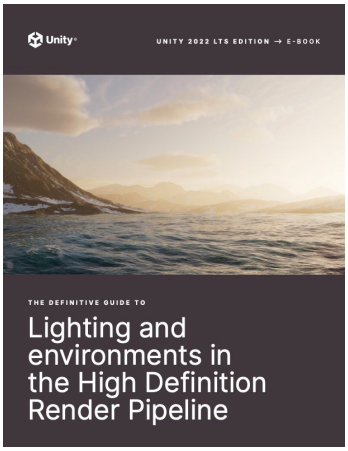
아티스트, 테크니컬 아티스트, 디자이너를 위한 Unity 툴셋과 URP 및 HDRP에 대해 자세히 알아보려면 다음 전자책을 참고하세요.



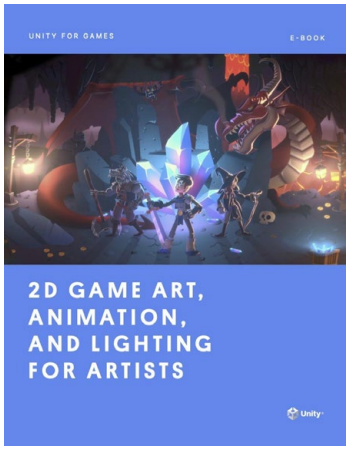
다운로드



다운로드



다운로드



다운로드

다운로드

다운로드

Unity로 애니메이션 임포트하기

많은 스튜디오에서 Autodesk Maya 또는 Blender 같은 소프트웨어를 사용하여 애니메이션을 제작한 다음 Unity로 임포트합니다.

전문적인 대규모 프로젝트에서는 흔히 움직임을 표현할 배우를 고용한 다음 모션 캡처를 이용해 콘텐츠에 활용할 사실적인 움직임을 구현합니다. 배우는 3D 골격의 주요 위치에 마커가 배치된 모션 캡처 슈트를 착용합니다. 고속 적외선 카메라가 배우의 움직임을 포착하고 Autodesk MotionBuilder 같은 소프트웨어로 데이터가 전달됩니다. 얼굴 모션도 같은 방식으로 캡처되며, 배우의 얼굴 전체에 배치된 마커가 사실적인 얼굴의 움직임을 기록합니다.

이 작업을 마치면 해당 애니메이션을 Unity에 임포트할 수 있습니다.

스톡홀름에 위치한 유니티 데모 팀은 모션 캡처를 사용하여 단편 애니메이션 아담(Adam), 더 헤러틱(The Heretic), 에너미즈(Enemies)를 제작했습니다.

게임을 포함한 실시간 3D 콘텐츠에서 인간의 얼굴과 몸을 사실적으로 움직이려면 모션 캡처가 필요한 경우가 많습니다.

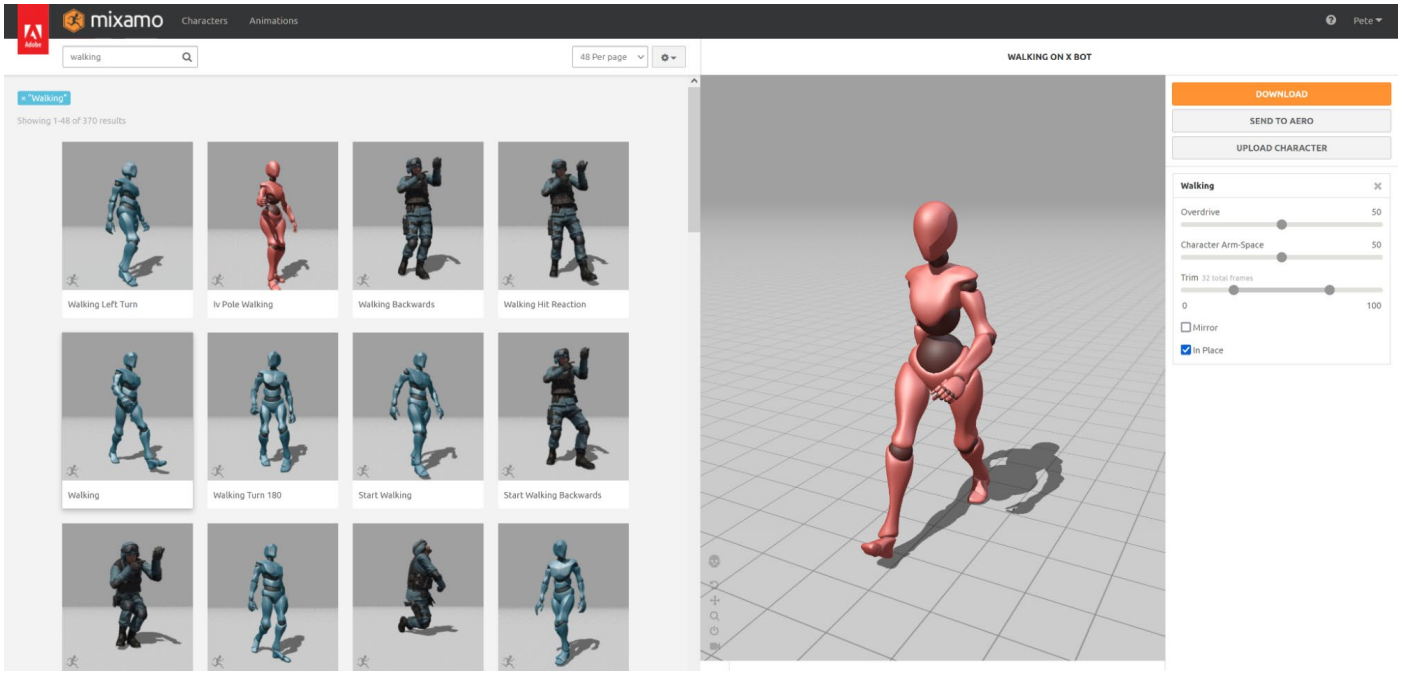
이 블로그 게시물에서 애니메이터 크라시미르 네체프스키는 유니티의 아담을 제작하며 진행한 사전 시각화, 모션 캡처, 애니메이션 작업 과정에 대해 설명합니다.

유니티 단편작 아담 의 제작 과정 중에 나오는 한 장면

Unity의 ARKit를 Apple iPhone과 함께 사용하여 얼굴 움직임과 표정을 캡처하여 캐릭터의 얼굴 릭과 입 움직임을 제어할 수도 있습니다. 이 방법이 사용된 예를 확인하려면 Realtime Rascals Unity 프로젝트를 살펴보세요.

실시간 얼굴 캡처로 캐릭터의 얼굴 제어

모션 라이브러리

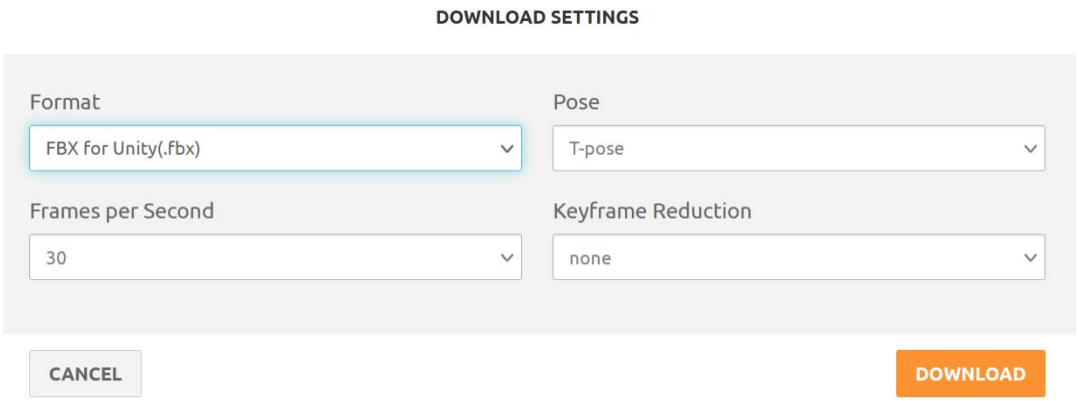


Adobe Mixamo 모션 라이브러리

모션 캡처 슈트나 모션 캡처 소프트웨어를 사용할 수 없다면 모션 캡처 라이브러리에서 사전 제작된 애니메이션 클립을 사용할 수 있습니다. 이러한 클립을 사용하면 시간을 절약하고 적은 예산으로 애니메이션을 제작할 수 있지만 실제 배우가 연기하는 것만큼의 정확도를 기대하기는 어렵습니다.

Adobe Mixamo와 Reallusion ActorCore는 수천 개의 작은 모션 클립을 제공하는 모션 라이브러리 웹사이트로, Unity 같이 원하는 3D 소프트웨어에 모션 클립을 다운로드할 수 있습니다.

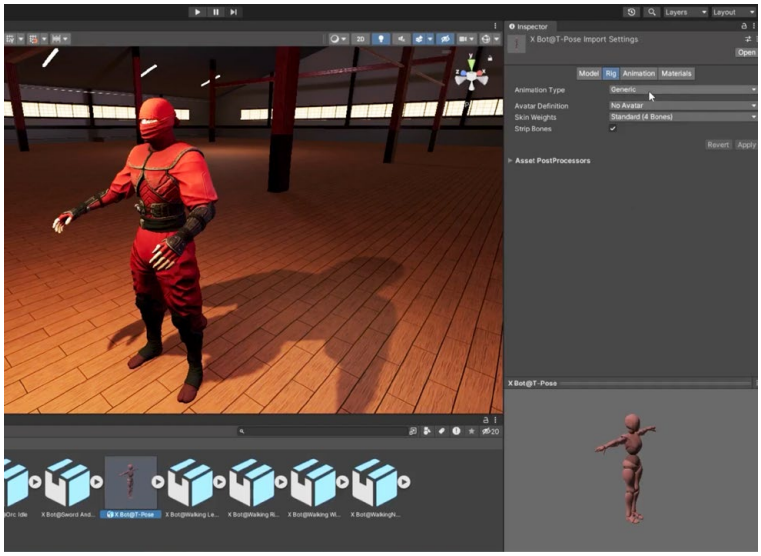
제공되는 다양한 무료 캐릭터 중에서 선택하는 것뿐 아니라 직접 제작한 캐릭터를 업로드할 수도 있습니다. 캐릭터의 골격 구조가 다르다면 휴머노이드 애니메이션 유형이 아닌 경우 애니메이션이 한 캐릭터에서는 재생되지만 다른 캐릭터에서는 재생되지 않을 수 있습니다. 휴머노이드 애니메이션 유형을 사용하려면 휴머노이드를 설정하는 데 필요한 T-포즈를 먼저 다운로드하세요(자세한 내용은 이 가이드의 후반부 섹션 참고).



Mixamo에서 모델을 다운로드하는 경우 Unity용 FBX 포맷과 T-포즈 설정을 선택해야 합니다.

Unity 에셋 스토어도 사전 제작된 애니메이션을 찾기 좋은 장소입니다. 애니메이션 카테고리를 검색한 다음 패키지 관리자를 통해 Unity에 패키지를 다운로드하세요. 그러면 애니메이션이 자동으로 설정되고 바로 사용할 수 있습니다.

사전 제작된 애니메이션을 사용하는 것 외에도 기성 툴을 다운로드하여 프로젝트의 품질을 향상할 수 있습니다. 위 이미지에 보이는 Sakura Rabbit의 캐릭터 및 애니메이션 작업에서는 Final IK, MagicaCloth, Face Capture 같은 Unity 에셋 스토어 패키지를 사용합니다.



휴머노이드 애니메이션 작업

이 동영상에서는 다운로드한 애니메이션을 Unity로 임포트하고 루트 모션을 사용하여 휴머노이드 애니메이션 유형으로 설정하는 방법을 설명합니다. 이외에도 다음과 같은 워크플로를 살펴봅니다.

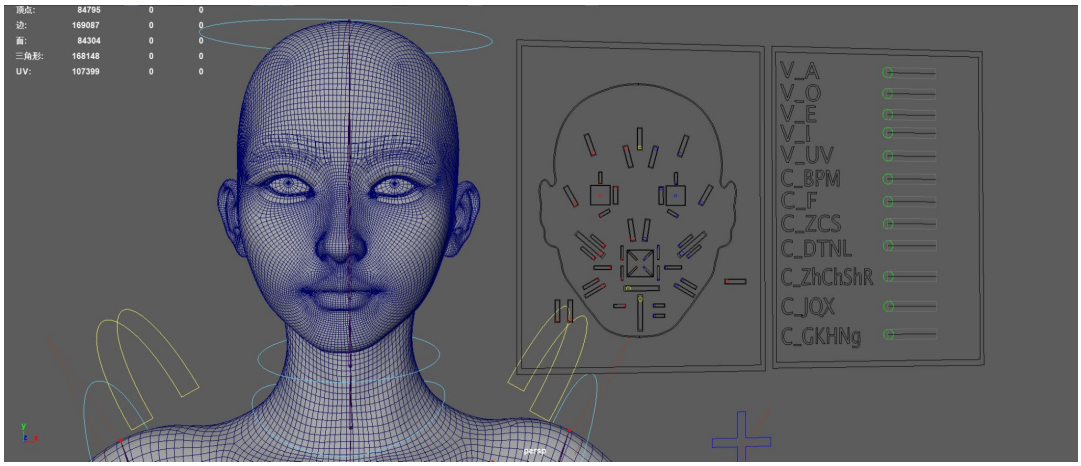
- IK를 포함한 애니메이션 리깅을 통해 애니메이션 수정
- 레이어를 사용하여 애니메이션 결합
- 속성을 설정하여 애니메이션 플로 제어
- 캐릭터 컨트롤러 설정
- AI 캐릭터 설정

동영상

Autodesk Maya의 애니메이션 익스포트하기

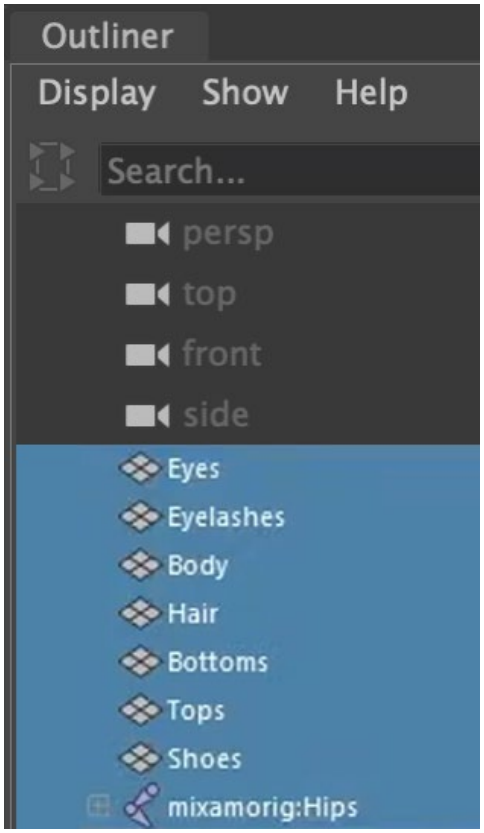
Maya IK 핸들은 Maya에서 Unity로 익스포트할 수 없습니다. Maya 릭으로 제어하는 모든 모션은 익스포트하기 전에 애니메이션 클립으로 베이크해야 합니다. Maya에서는 Send to Unity 옵션을 통해 이를 자동으로 처리할 수 있습니다.

하지만 자동 익스포트 옵션이 항상 적합하지는 않을 수 있습니다. 대규모 스튜디오에서는 테크니컬 아티스트가 스크립트를 활용하여 팀에 가장 적합한 방법을 사용하는 커스텀 익스포트 툴을 제작하기도 합니다.



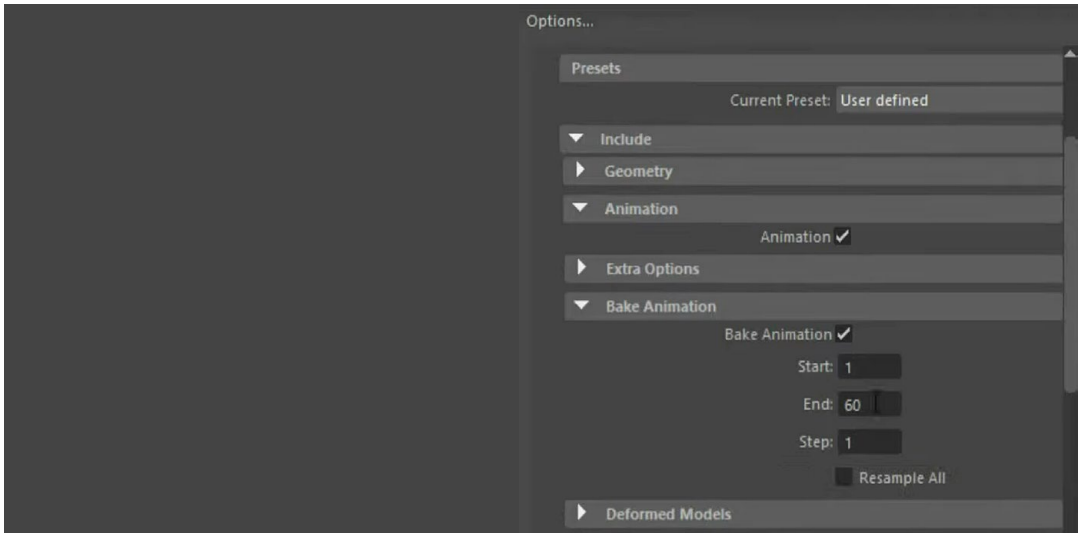
Sakura Rabbit이 제작한 캐릭터의 Maya 리깅 설정입니다.

보다 세밀한 제어를 위해 수동으로 캐릭터를 익스포트하는 방법은 다음과 같습니다.



Maya에서 전체 캐릭터 선택

1. 익스포트할 캐릭터를 선택합니다.
2. Outliner에서 고관절을 클릭하여 골격의 루트 노드를 선택합니다.
3. **Select > Hierarchy**로 이동하여 연결된 모든 뼈대를 선택합니다.
4. Shift 키를 누른 상태에서 캐릭터의 모든 메시 오브젝트를 선택합니다.
5. **File > Export Selected**를 선택합니다. FBX 파일로 익스포트합니다.
6. 이제 애니메이션을 익스포트할지 옵션에서 선택할 수 있습니다. 선택하지 않으면 모델만 익스포트합니다. T-포즈 캐릭터인 경우 Unity에서 휴머노이드 애니메이션 유형을 설정하기 쉽습니다.
7. 애니메이션을 익스포트하도록 선택한 경우 애니메이션을 베이크하는 옵션도 선택해야 합니다. 시작 프레임과 종료 프레임을 설정할 수 있습니다. 모든 모션이 키프레임 애니메이션 파일로 베이크되어 3D 메시와 함께 익스포트됩니다.

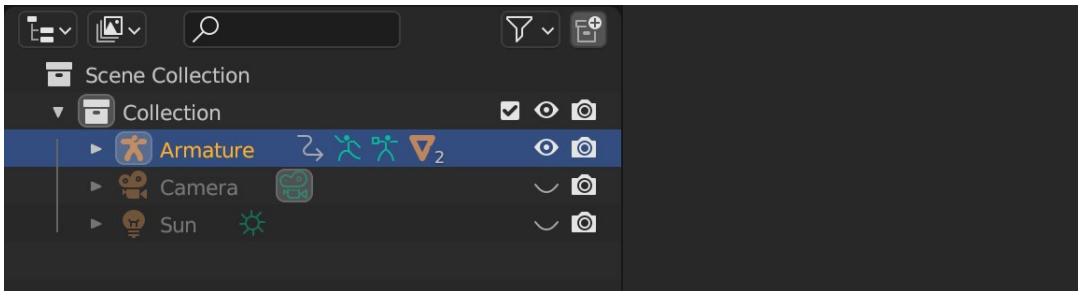


Maya에서 익스포트 시 애니메이션 베이크하기

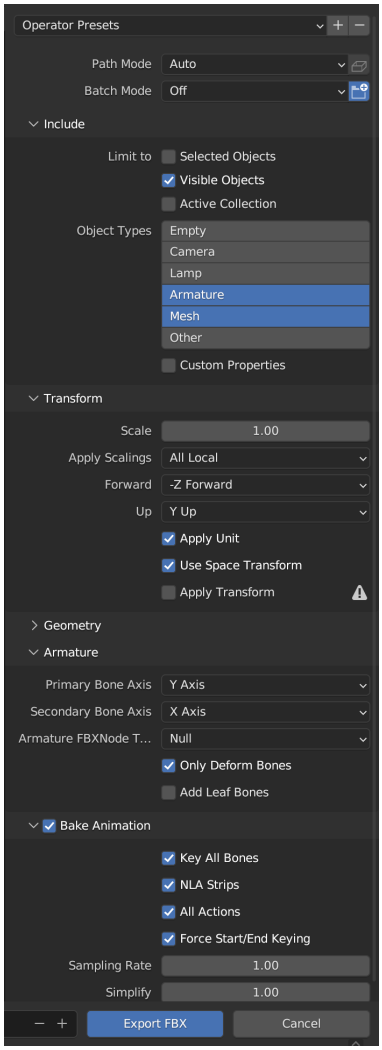
Blender의 애니메이션 익스포트하기

Blender에서 익스포트하는 방법은 다음과 같습니다.

1. 익스포트할 대상이 아닌 오브젝트는 옆에 있는 가시성(눈 모양) 아이콘을 클릭하여 비활성화합니다. 익스포트할 캐릭터만 보이도록 설정해야 합니다.



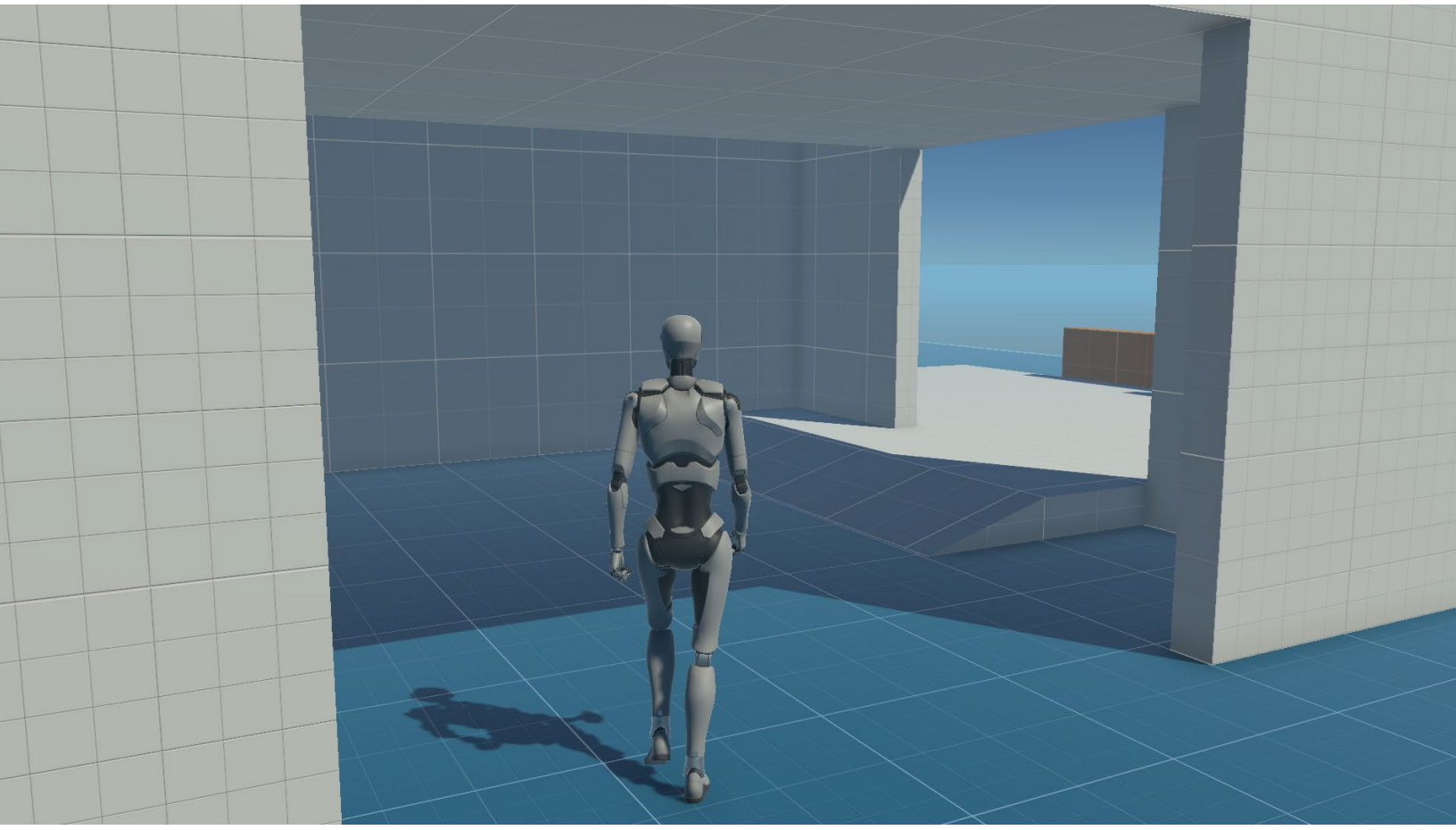
익스포트하지 않을 항목은 표시 안 함



애니메이션화된 캐릭터에 대한 Blender의 익스포트 옵션

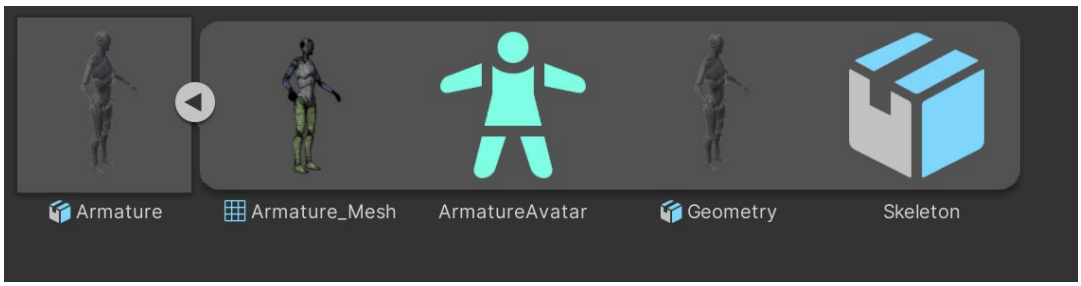
2. Armature를 선택하되, 컨트롤 릭은 Blender에서 Unity로 익스포트할 수 없으므로 선택하지 않습니다.
3. **File > Export > FBX**로 이동합니다.
4. FBX 익스포트 옵션에서 Visible Only를 선택합니다. 이렇게 하면 컨트롤 릭이나 다른 항목이 파일에 익스포트되지 않습니다. 오브젝트 유형에서 Armature와 Mesh만 선택합니다(Shift 키를 누른 상태에서 여러 옵션 선택).
5. **Armature** 탭에서 **Only Deform Bones**를 선택하고 **Add Leaf Bones**는 선택하지 않습니다. Deform Bones란 메시를 변형하는 골격의 뼈대를 의미합니다. **Bake Animation**을 선택하지 않으면 메시만 익스포트합니다. Bake Animation을 선택하면 컨트롤 릭에서 설정한 동작을 포함한 모든 애니메이션이 키프레임 애니메이션 클립으로 베이크되어 함께 익스포트됩니다. 비선형 에디터의 모든 애니메이션 스트립이 별도의 임베디드 애니메이션 클립으로 익스포트됩니다.

Unity로 애니메이션 임포트하기



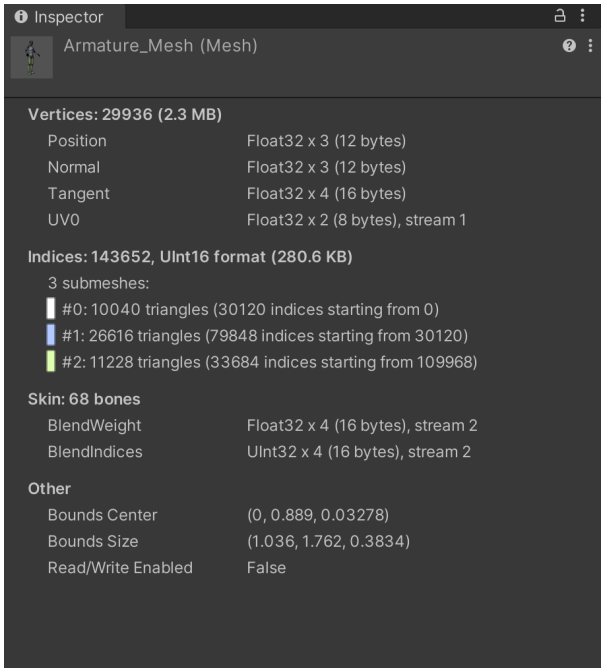
Unity에 임포트된 애니메이션화된 캐릭터

임베디드 애니메이션이 있는 3D 모델의 포맷으로 FBX가 주로 사용됩니다. Unity에 FBX 파일을 임포트하려면 컴퓨터의 파일 브라우저에서 에디터의 프로젝트(Project) 창으로 직접 드래그합니다.



FBX 파일의 구조(Unity의 Starter Assets - ThirdPerson CharacterController 패키지에서 가져옴)

FBX 파일에는 메시, 지오메트리, 머티리얼, 골격, 아바타, 애니메이션을 비롯한 다양한 임베디드 컴포넌트와 프로퍼티가 있습니다.



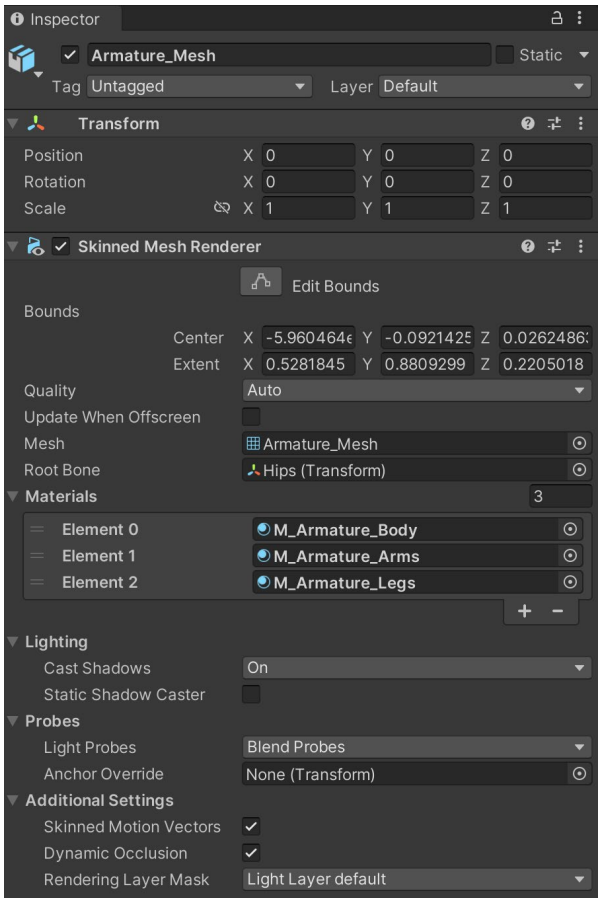
FBX 파일의 메시 데이터

메시

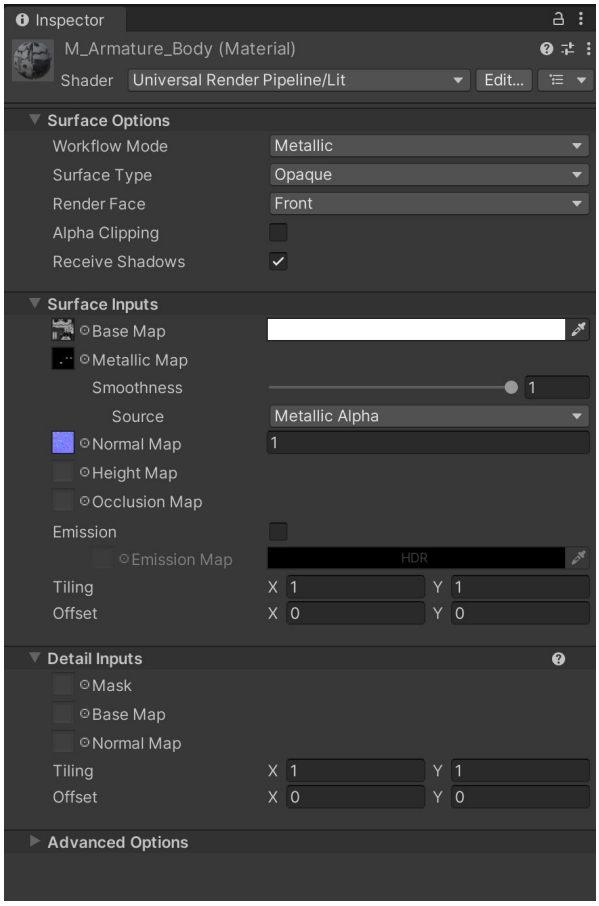
메시는 오브젝트의 셰이프를 정의하는 3D 모델 데이터입니다. Unity에서 메시 에셋에는 모델의 모든 버텍스, 노멀, 블렌드 셰이프, UV 데이터가 담겨 있습니다.

지오메트리

지오메트리는 메시를 기반으로 하는 물리적 모델입니다. 지오메트리는 씬에서 표현되는 방법과 스킨이 골격에 결합되는 방법을 정의하는 Skinned Mesh Renderer 컴포넌트의 영향으로 골격의 뼈대가 움직이는 것에 따라 움직이게 됩니다. Skinned Mesh Renderer 컴포넌트를 통해 지오메트리에 머티리얼을 할당할 수 있습니다.



FBX 파일의 지오메트리



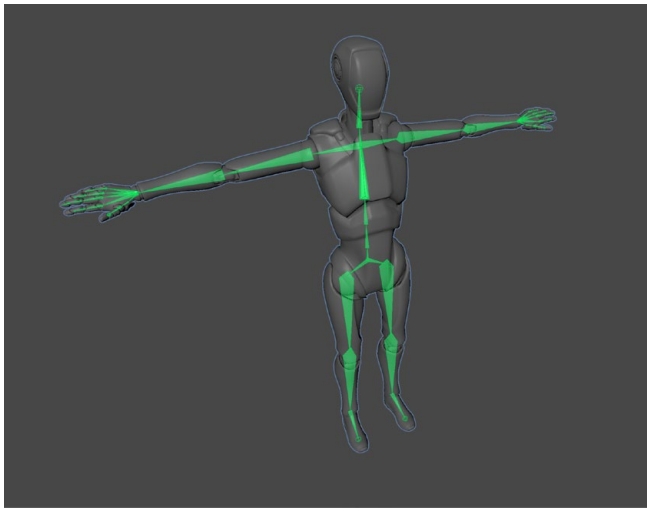
FBX 파일의 머티리얼 설정

머티리얼

머티리얼은 오브젝트 표면이 렌더링 시 어떻게 보이는지 정의하는 에셋으로, 지오메트리에 적용된 텍스처를 Unity 셰이더로 결합하여 씬에서 모델이 어떻게 표현되는지 정의합니다.

골격

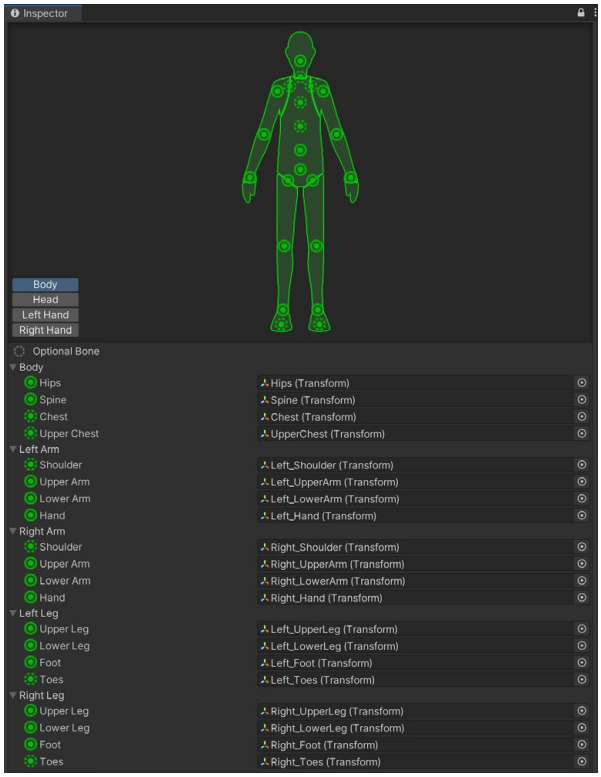
골격은 뼈대의 모음(이족 보행 캐릭터의 경우 최소 15개)으로, 메시와 결합되어 씬 내 캐릭터의 움직임 애니메이션을 구현합니다. 골격은 씬(Scene) 뷰 및 게임(Game) 뷰에서 기본적으로 보이지 않으며, 계층 구조(Hierarchy)에서 액세스할 수 있습니다. 골격을 시각화하려면 이 전자책 뒷부분의 애니메이션 리깅 섹션에서 설명하는 Bone Renderer를 사용하세요.



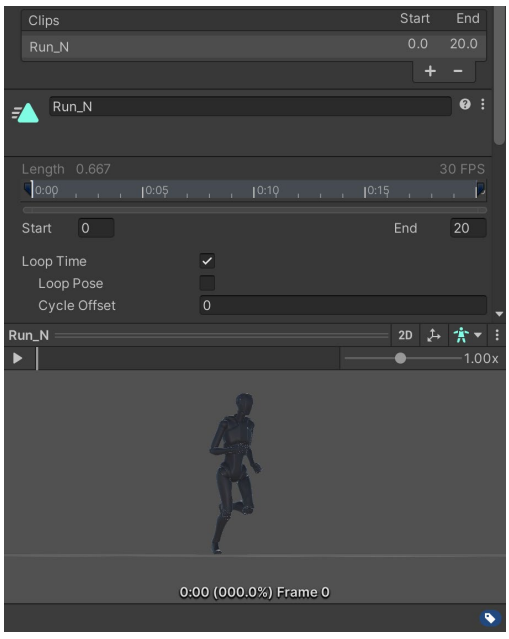
FBX 파일 애니메이션화에 사용되는 골격

아바타

아바타 는 캐릭터의 뼈대 레이아웃을 정의하며, Unity 애니메이터 시스템으로 캐릭터 또는 오브젝트에 어떤 애니메이션을 추가할 수 있는지 정의합니다. 제네릭 아바타는 이족 보행 구조가 아닌 동물과 생물에 사용할 수 있습니다.



FBX 파일의 뼈대 이름을 정의하는 휴머노이드 아바타 레이아웃

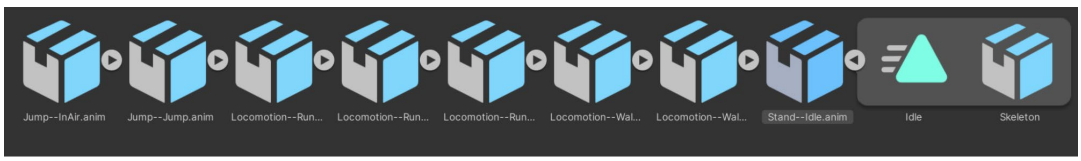


FBX 파일에 내장된 애니메이션 클립

애니메이션

FBX 파일에는 내장된 애니메이션 클립 여러 개를 저장할 수 있습니다. 각 애니메이션 클립에는 캐릭터의 골격 애니메이션을 표현하는 키프레임 모션 데이터가 저장됩니다.

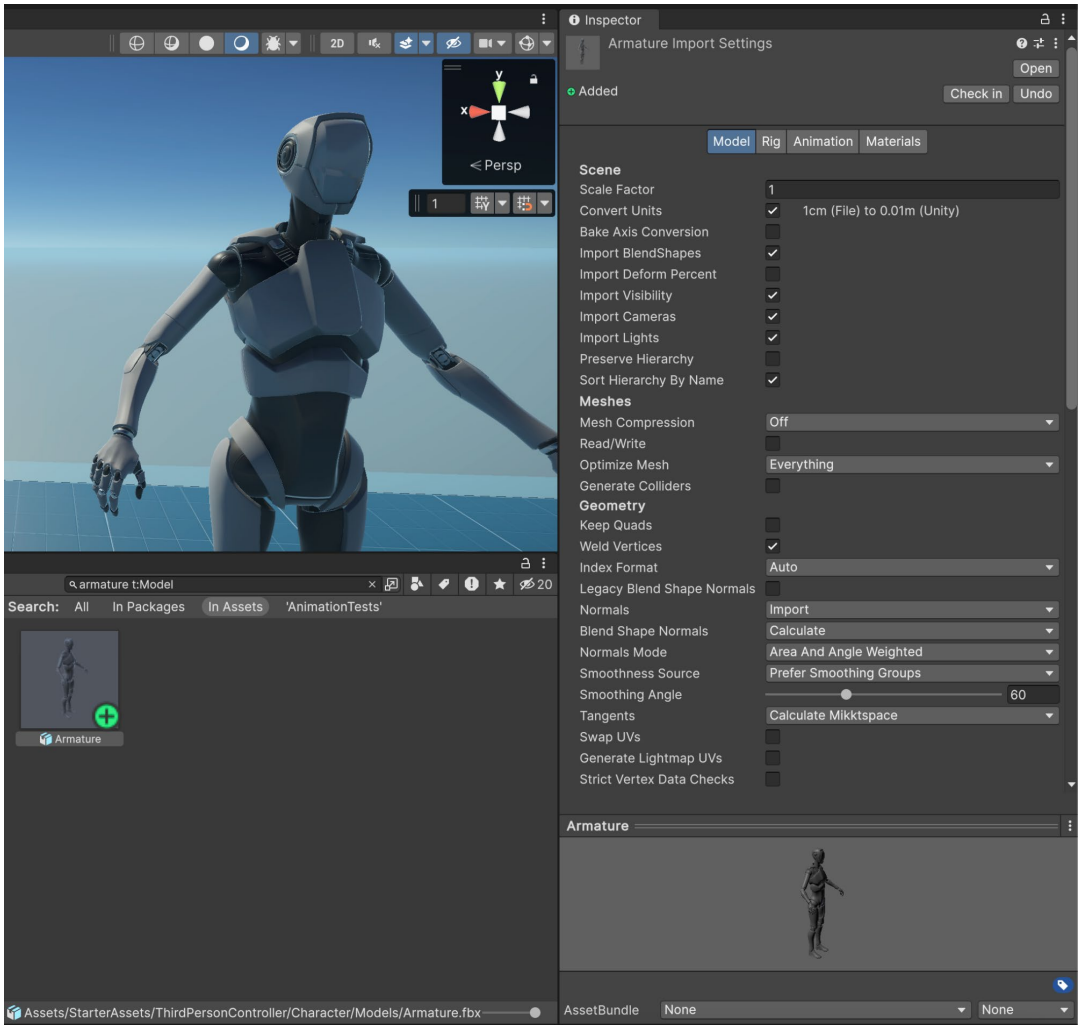
지오메트리 없이 애니메이션만 포함된 FBX 파일을 익스포트하면 골격과 애니메이션 클립이 포함되며, 삼각형 아이콘으로 표시됩니다.



애니메이션 전용 FBX 파일

FBX 임포트 루트 오브젝트

FBX 임포트 루트 오브젝트는 FBX 파일의 설정이 포함된 메인 FBX 오브젝트입니다.



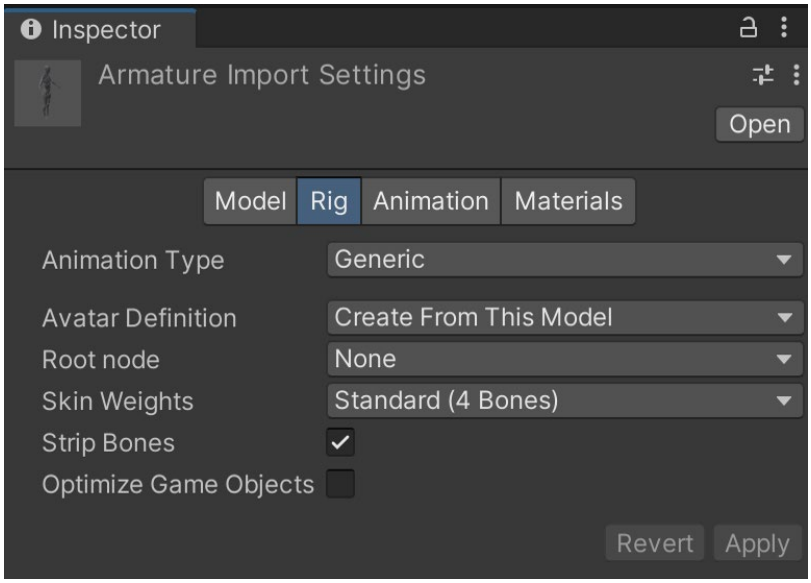
FBX 파일의 임포트 설정

Unity 매뉴얼의 더 많은 자료:

- Model 탭
- Rig 탭
- Animation 탭
- Materials 탭

제네릭 애니메이션 유형

Unity는 제네릭 및 휴머노이드라는 두 가지 주요 애니메이션 유형을 사용하여 캐릭터 애니메이션을 제어합니다. 휴머노이드 유형은 모든 이족 보행 캐릭터에 적용할 수 있는 보편적인 구조입니다. 제네릭 유형은 각 캐릭터의 골격 구조별로 다릅니다. 애니메이션 커브에는 위치, 회전, 스케일, 정규화된 근육 공간이 있습니다.

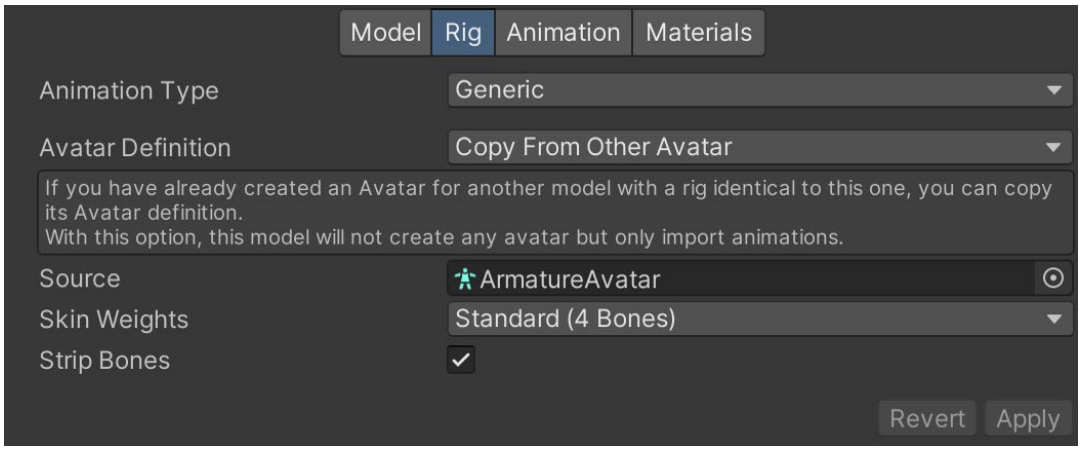


FBX 파일의 릭 설정

제네릭 애니메이션 유형은 순운동학(FK)을 사용하여 Unity에서 직접 애니메이션 키프레임을 설정하려는 경우에 사용할 수 있으며, 골격 구조가 서로 다른 캐릭터 간에 애니메이션을 공유할 필요가 없을 때 사용합니다.

자체 FBX 파일을 임포트하면 기본적으로 **Generic**으로 설정됩니다. 이 애니메이션 유형은 특정 캐릭터와 그 골격 레이아웃에 따라 고유하게 나타납니다. 제네릭 애니메이션 유형을 사용할 때 골격 구조가 다른 캐릭터의 애니메이션은 이 캐릭터에서 작동하지 않을 수 있지만, 골격 레이아웃이 같은 캐릭터 간에는 제네릭 애니메이션을 공유할 수 있습니다.

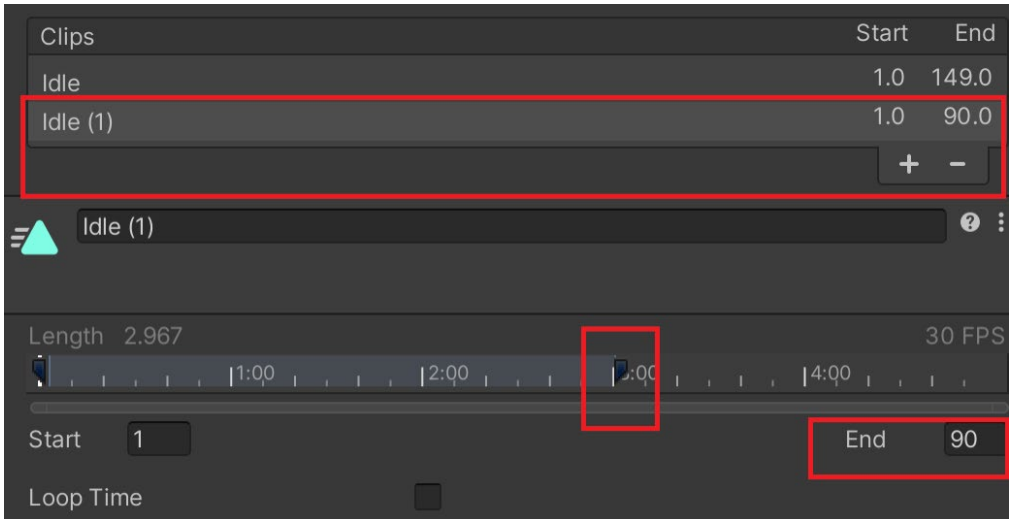
FBX 파일의 Rig 탭에서 드롭다운 메뉴를 사용해 언제든지 애니메이션 유형을 변경할 수 있습니다. 이 모델에 아바타가 없다면 생성할 수도 있습니다. T-포즈 또는 A-포즈의 모델에서 아바타를 생성하는 것이 가장 좋습니다.



애니메이션 전용 FBX 파일 릭 설정

애니메이션 전용 FBX 파일도 기본적으로 제네릭 애니메이션 유형을 사용합니다. 애니메이션이 캐릭터의 골격에 바르게 매치되려면 해당 파일에서 FBX 캐릭터 파일의 아바타를 참조해야 합니다.

FBX 임포트 설정의 **Animation** 탭에서 클립 섹션 아래의 **+** 아이콘을 클릭하여 애니메이션 클립을 정의할 수 있습니다. 한 클립에 여러 개의 모션이 있기도 하며, 이런 경우에는 각 모션을 개별 클립으로 나눠야 합니다. 정의된 클립이 없다면 첫 클립의 이름이 Take01으로 설정됩니다.

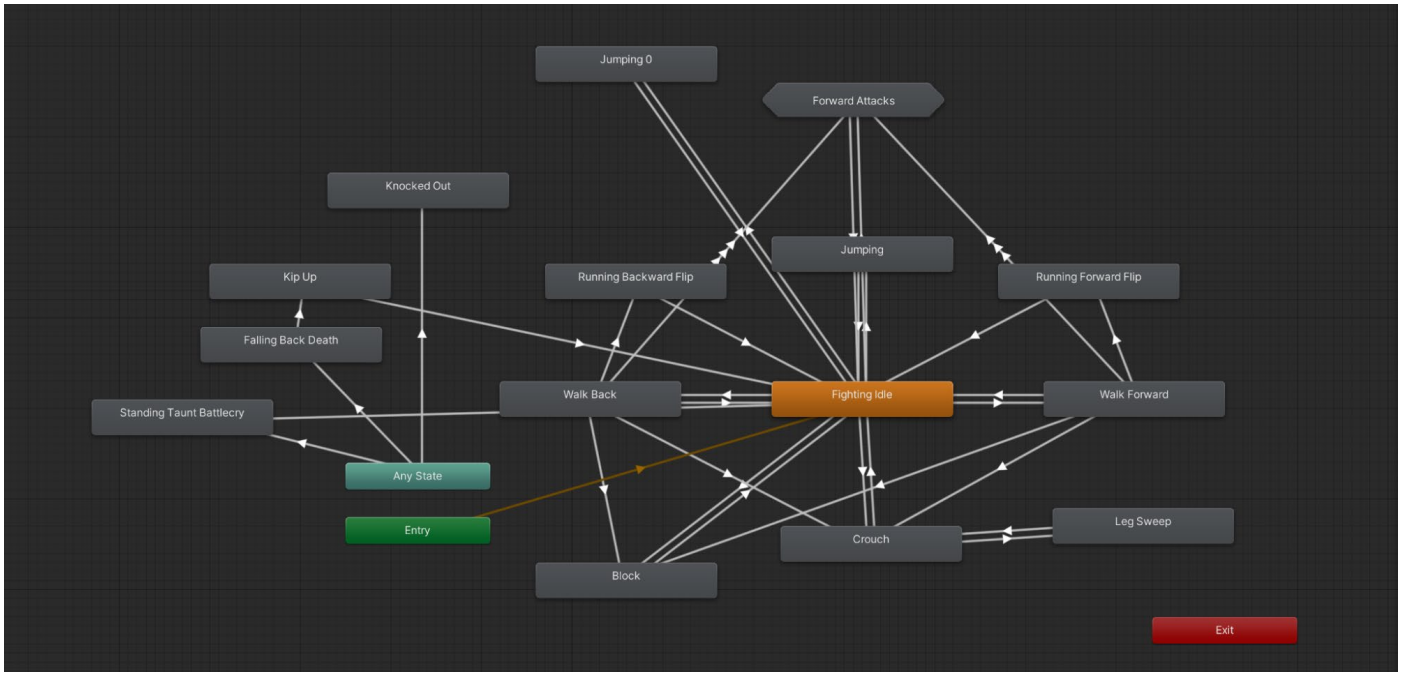


애니메이션 길이 복제 및 조정

시작/종료 마커를 조정하거나 **Start** 또는 **End** 슬롯에 프레임 번호를 입력하여 클립의 타이밍을 변경할 수 있습니다. **Apply**를 클릭하면 애니메이션 클립을 캐릭터 프리팹으로 임포트합니다.

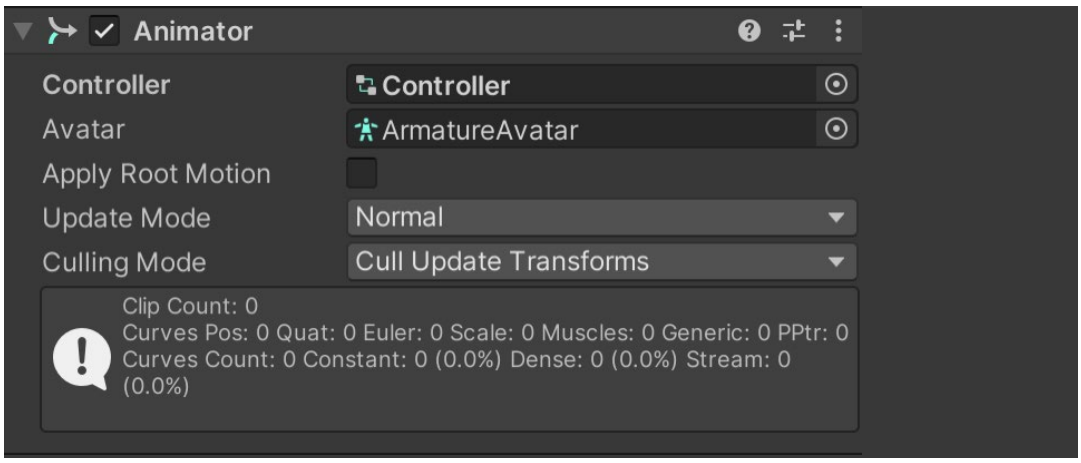
클립이 반복 재생되도록 설정하거나 반복 재생이 원활하지 않다면 사이클 오프셋을 설정할 수도 있습니다. 사이클 오프셋을 몇 프레임 조정하면 클립의 첫 키프레임과 마지막 키프레임을 잘 연결할 수 있습니다.

Animator 컨트롤러



Unity의 애니메이션 상태 머신

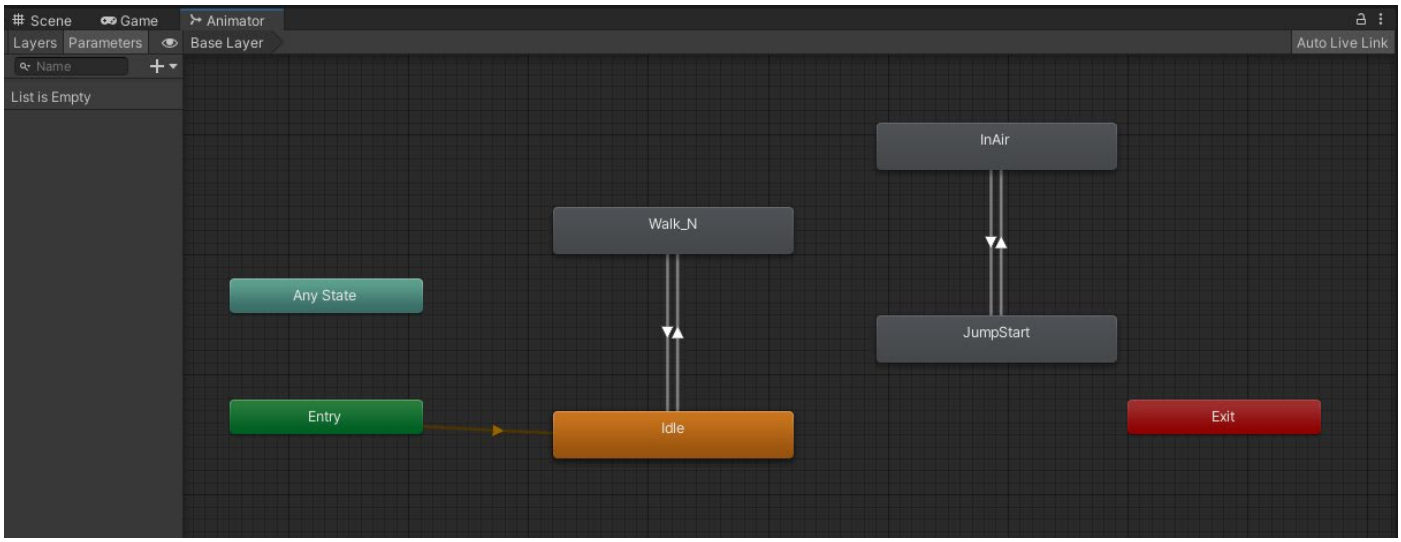
Animator 컨트롤러는 캐릭터의 모든 애니메이션을 저장하고 실행하는 역할을 담당합니다. Animator 컨트롤러를 생성하려면 프로젝트 탭을 오른쪽 클릭하고 **Create > Animator Controller**를 선택합니다. 캐릭터에 Animator 컨트롤러를 처리할 Animator 컴포넌트가 있어야 합니다.



Animator 컴포넌트

더 빠른 방법은 프로젝트 뷰에서 계층 구조의 캐릭터 프리팹으로 클립을 드래그하는 것입니다. 이렇게 하면 캐릭터에 Animator 컴포넌트가 추가되고 Animator 컨트롤러에 애니메이션 클립이 추가됩니다.

컨트롤러에 액세스하려면 **Window > Animations > Animator**를 통해 Animator 창을 열거나 프로젝트 뷰에서 Animator 컨트롤러를 더블 클릭합니다. 이제 애니메이션 클립을 드래그하여 캐릭터에 할당할 수 있습니다. 캐릭터에 설정된 애니메이션 유형에 맞는 애니메이션만 할당해야 합니다. 제네릭 캐릭터에는 제네릭 애니메이션을, 휴머노이드 캐릭터에는 휴머노이드 애니메이션을 할당합니다.

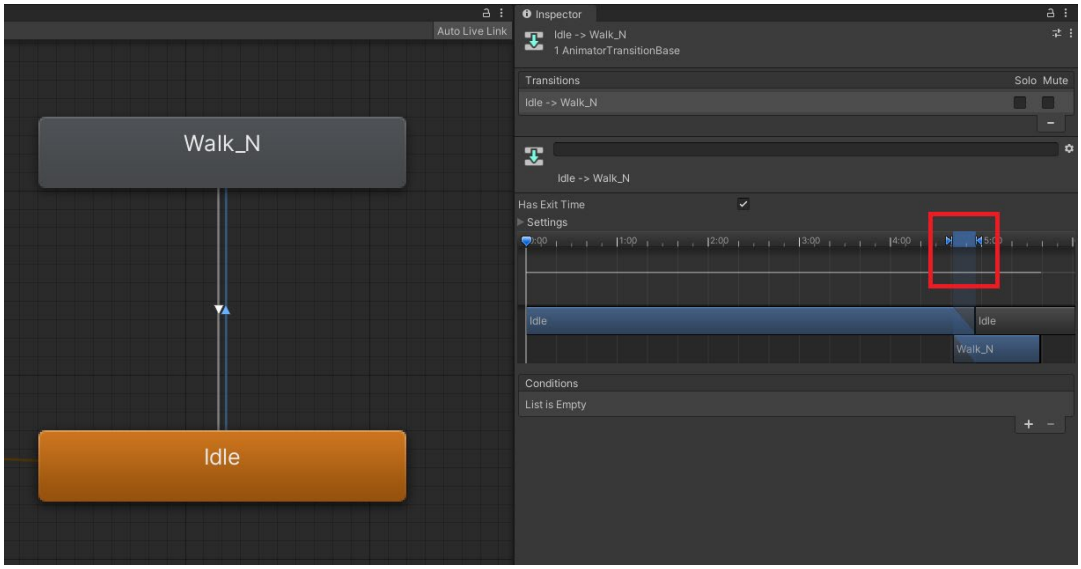


Animator 창 레이아웃

위 Animator 창의 스크린샷에 보이는 주황색 블록은 애니메이션 상태 **노드**입니다. 이 노드는 가장 먼저 재생할 기본 애니메이션을 나타냅니다. 한 노드에서 다른 노드로 이동하는 전환을 추가하려면 노드를 오른쪽 클릭하고 **Make Transition**을 선택한 다음 원하는 애니메이션 노드를 선택합니다.

위 예시의 전환은 **Idle**과 **Walk_N** 사이를 계속 왕복하며 루프를 구성합니다. 점프 동작을 위한 다른 2개의 노드는 Idle이나 Walk_N으로 곧바로 전환되지 않으므로 재생되지 않습니다. 애니메이션이 활성화되려면 주황색 노드의 기본 상태에 연결된 노드에서 전환되거나 **Any State** 노드에서 전환되어야 합니다.

전환 선을 클릭하면 전환이 일어나는 조건이나 방법 등 해당 전환의 설정을 확인할 수 있습니다.



두 애니메이션 간의 전환을 블렌딩하는 시간

위 이미지에서 빨간색 상자로 강조된 작은 파란색 섹션은 두 애니메이션 간의 블렌딩 시간을 나타냅니다. 블렌딩 시간이 길수록 전환이 부드럽게 실행됩니다. 이렇게 하면 한 애니메이션에서 다음 애니메이션으로 전환할 때 부자연스러운 움직임을 방지할 수 있습니다.

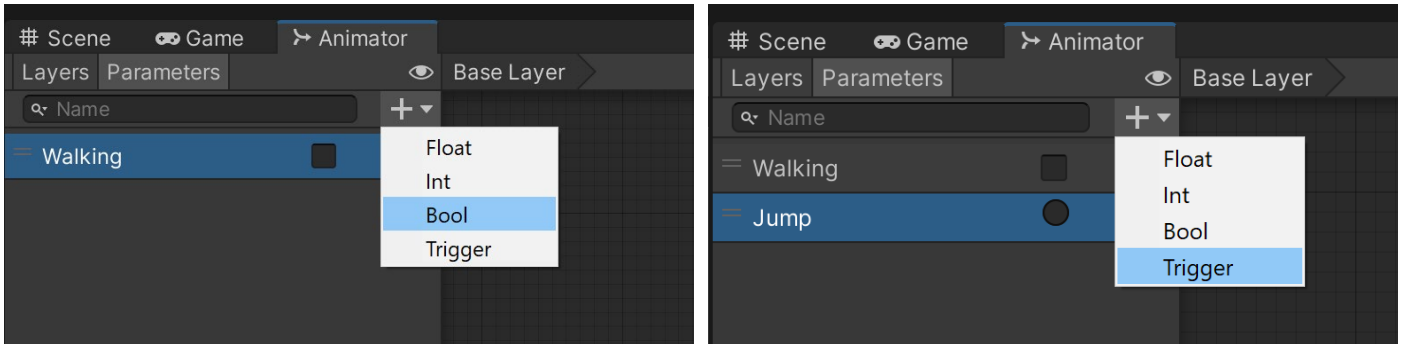
양쪽 파란색 화살표를 드래그하여 두 애니메이션 간의 블렌딩 시간을 늘리거나 줄일 수 있습니다.

Has Exit Time 옵션을 선택하면 첫 애니메이션이 완전히 재생된 후에야 다음 애니메이션으로 전환됩니다. 설정된 조건이 없다면 이 옵션을 선택합니다.

파라미터

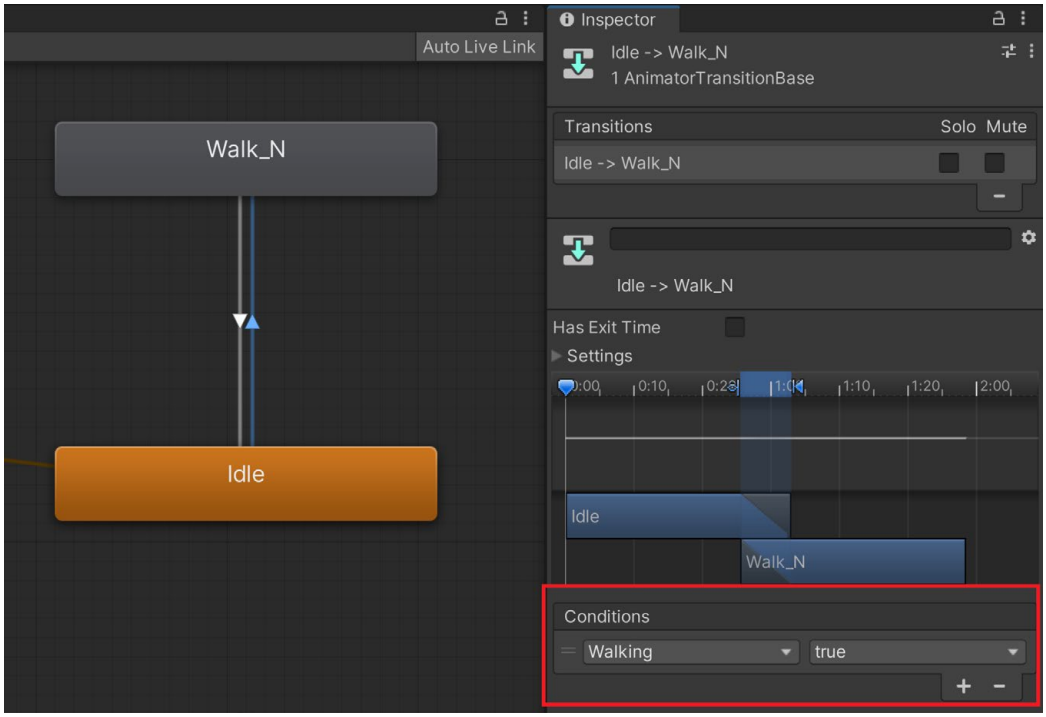
플레이어가 버튼을 누르는 행동 등으로 Animator 컨트롤러와 상호 작용하여 애니메이션을 실행할 수 있어야 합니다. 상호 작용을 위해 컨트롤러에 파라미터가 필요합니다. 파라미터는 애니메이션 전환에서 조건으로 설정할 수 있는 변수 입력입니다.

+ 아이콘을 클릭하여 Walking 파라미터에 Bool을 추가합니다. **Jump**라는 **Trigger** 파라미터도 설정합니다.



Animator 창에서 파라미터 생성

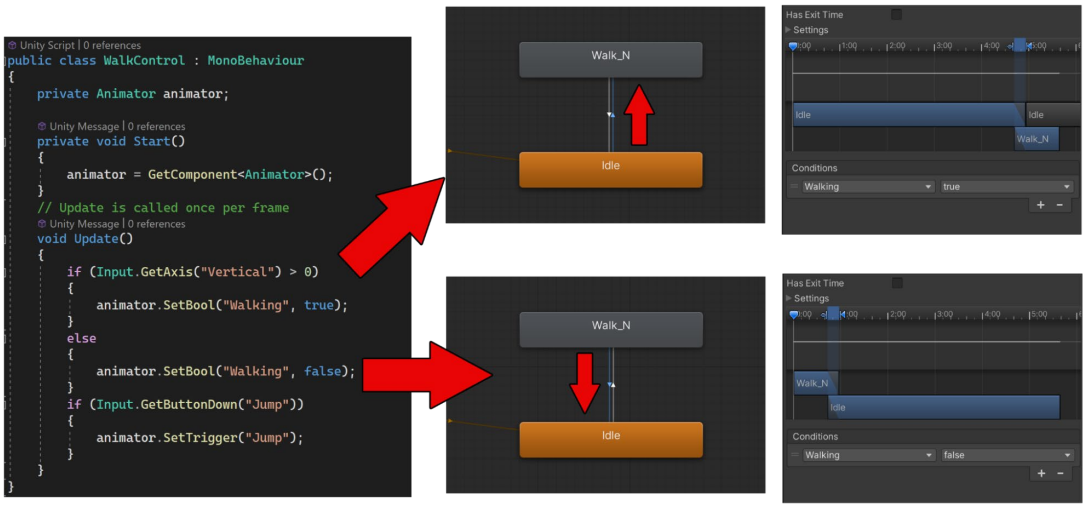
Idle에서 Walk_N으로 전환할 때, **Walking** 조건을 **true**로 설정합니다. **Has Exit Time** 옵션을 선택하지 않으면 Idle 애니메이션이 완료될 때까지 기다리지 않고 바로 전환됩니다.



전환 발생 시기를 제어하기 위해 Walking 파라미터에서 부울 조건을 설정합니다.

Walk_N에서 다시 Idle로 전환할 때, **Walking** 조건을 **false**로 설정합니다.

캐릭터의 파라미터를 제어하려면 C# 스크립트가 필요합니다.



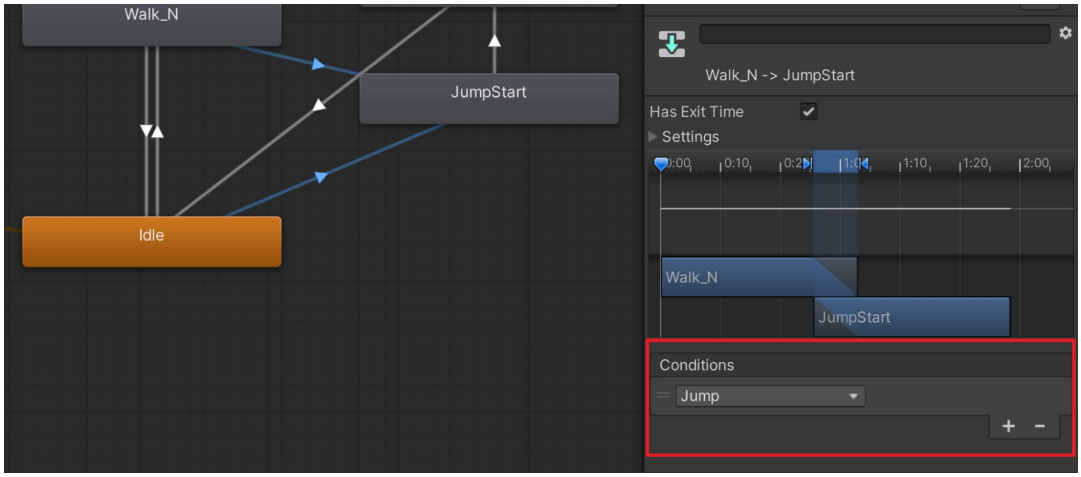
스크립트를 통해 애니메이터의 두 노드에서 전환을 제어하는 방법

스크립트의 Start()에서 animator = GetComponent<Animator>();를 통해 캐릭터의 Animator 컴포넌트를 가져와 제어합니다. Update 메서드에서는 플레이어가 세로 입력인 W 또는 위쪽 화살표 키를 누를 때까지 기다립니다. 키가 눌리면 애니메이터의 Walking 부울이 **true**로 설정됩니다. 그러면 Idle에서 Walk_N으로 전환됩니다.

플레이어가 위쪽 키를 누르고 있는 동안 Walk_N 애니메이션을 계속 재생합니다. 플레이어가 키 입력을 멈추면 애니메이터의 Walking 부울이 **false**로 설정됩니다. Walk_N에서 Idle로 전환되고 플레이어가 위쪽 화살표 키를 다시 누를 때까지 Idle 애니메이션이 재생됩니다.

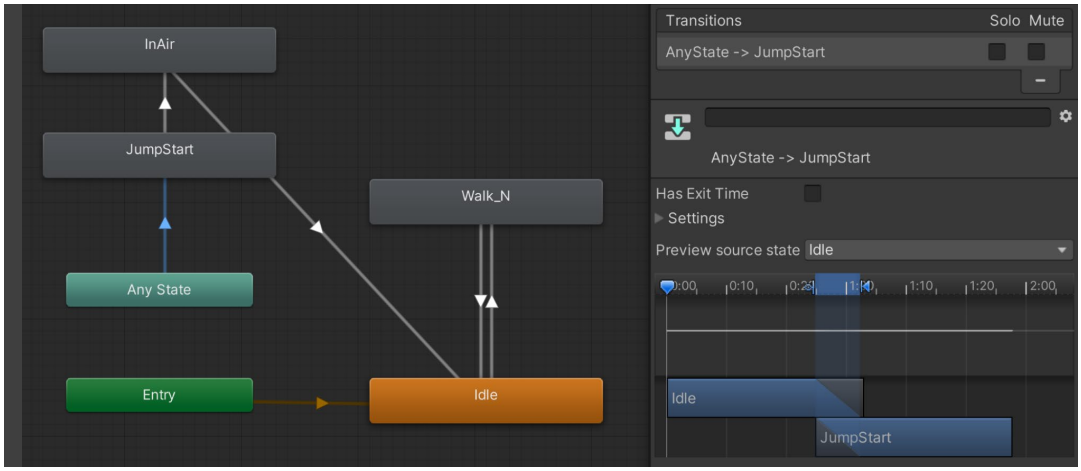
Update 메서드도 플레이어가 점프 키(기본적으로 스페이스바)를 누를 때까지 기다립니다. 점프 키를 누르면 애니메이터의 Jump 트리거가 활성화되어 스페이스바를 누를 때마다 점프 애니메이션을 재생합니다.

대기(idle) 상태 및 걷기 상태에서 모두 점프할 수 있어야 하므로 점프 애니메이션 전환도 2개이며, 두 전환의 조건은 **Jump**로 설정합니다.



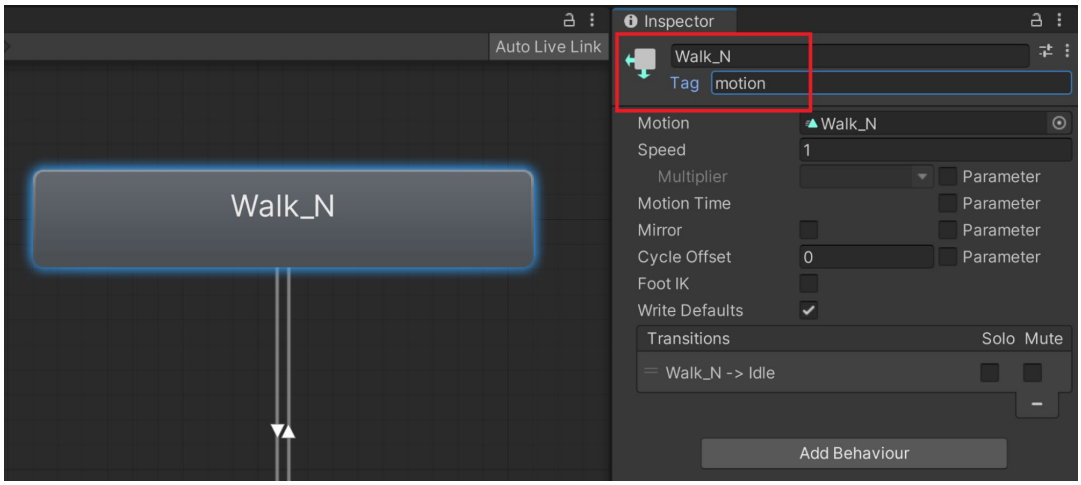
트리거 파라미터 조건이 Walk_N 및 Idle 양측에서 점프가 발생하는 시점을 제어하도록 설정됩니다.

또 다른 방법은 **Any State** 노드에서 전환하도록 하는 것입니다. 하지만 이렇게 하면 전체 애니메이터의 클립에서 호출될 수 있다는 점에 유의해야 합니다. 애니메이션 루프를 유지하기 위해 완료된 후에는 Idle로 다시 전환합니다.



모든 애니메이션 노드의 전환에 Any State 노드 사용

애니메이션 노드를 클릭하여 재생 속도를 느리게 또는 빠르게 조정할 수 있습니다. **Tag** 필드에 애니메이션 태그를 할당하여 코드 작성 시 유용하게 활용할 수도 있습니다. 아래 예시에서 Walk_N 애니메이션에는 motion 태그가 있지만 Idle 애니메이션에는 태그가 없습니다. 이렇게 하면 코드에서 캐릭터가 움직이는 시점을 식별할 수 있습니다.



태그 설정을 비롯한 애니메이션 상태 설정

스크립트에서 애니메이션을 모니터링하려면 AnimatorStateInfo 구조체를 사용합니다. 프레임마다 애니메이터의 현재 동작을 스크립트에서 확인할 수 있습니다. 아래 예시의 Update 메서드에서는 기본 레이어인 0번 레이어에서 애니메이터의 정보를 가져옵니다.

motion 태그가 달린 애니메이션 클립이 재생되는 경우 transform.Translate 커맨드를 통해 캐릭터를 앞으로 움직입니다.

```
Unity Script (1 asset reference) | 0 references
public class WalkControl : MonoBehaviour
{
    private Animator animator;
    private AnimatorStateInfo info;

    Unity Message | 0 references
    private void Start()
    {
        animator = GetComponent<Animator>();
    }
    // Update is called once per frame
    Unity Message | 0 references
    void Update()
    {
        info = animator.GetCurrentAnimatorStateInfo(0);
        if (info.IsTag("motion"))
        {
            transform.Translate(transform.forward * 2 * Time.deltaTime);
        }
```

코드를 통한 애니메이터 모니터링

AnimatorStateInfo 구조체를 사용하면 애니메이션이 재생되거나 전환되는 시점을 파악하여 Animator 컴포넌트에 맞춰 코드를 실행할 수 있습니다.

레이어

레이어를 사용하면 캐릭터의 애니메이션을 여러 세트로 나눌 수도 있습니다. 캐릭터의 신체 부위마다 다른 애니메이션을 적용할 수 있으므로 여러 애니메이션을 결합해 새로운 독특한 애니메이션을 만들 수 있습니다.

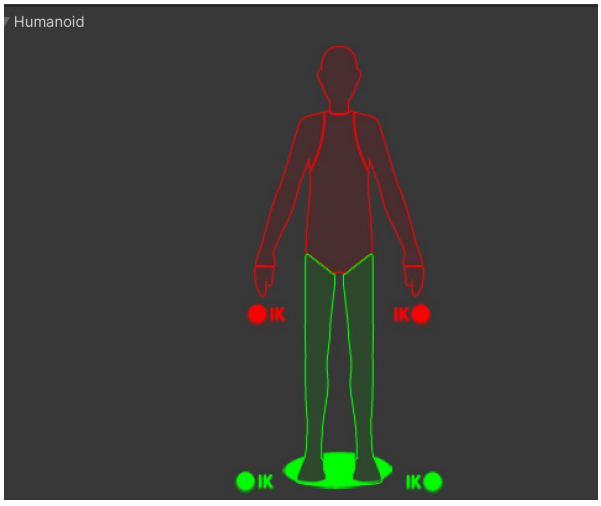
기본 걷기 애니메이션은 애니메이터의 0번 레이어인 Base Layer에 설정되어 있습니다.

기본 걷기 애니메이션

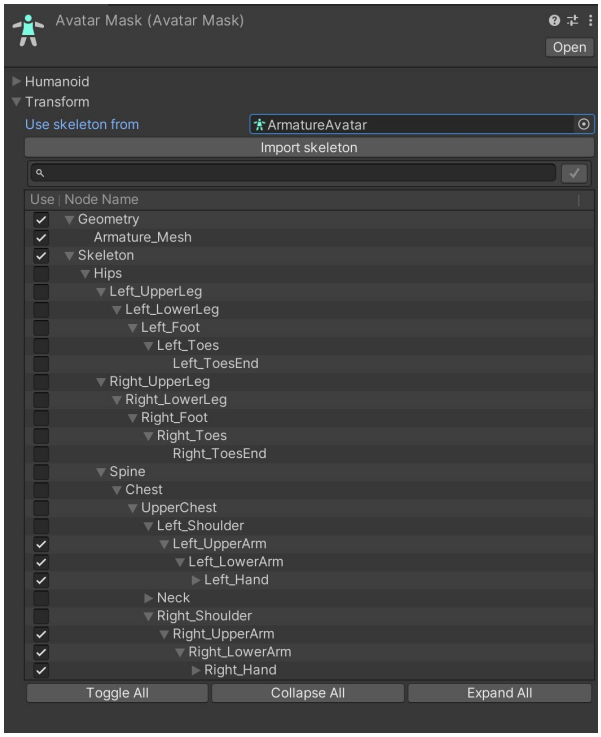
아바타 마스크를 사용하면 신체의 어느 부위에 기본 걷기 애니메이션을 적용하고 어느 부위에 다른 애니메이션을 적용할지 정의할 수 있습니다. 아바타 마스크를 레이어 시스템에 적용하면 특정 애니메이션에 영향을 받는 특정 신체 부위를 제외할 수 있습니다. 모션을 결합할 수도 있습니다. 이를테면 상체에 어떤 동작을 적용하고 하체에는 다른 동작을 적용하는 것이 가능합니다.

아바타 마스크를 생성하려면 프로젝트 창에서 오른쪽 클릭하고 **Create > Avatar Mask**를 선택합니다. 인스펙터(Inspector)에서 캐릭터에 사용 중인 애니메이션 유형에 따라 두 가지 유형 중 선택할 수 있습니다.

휴머노이드 유형을 선택하면 캐릭터에 대한 시각적 가이드를 확인할 수 있으며, 캐릭터가 휴머노이드 애니메이션 유형일 때만 사용할 수 있습니다. Humanoid 섹션에서 신체 부위를 클릭해 제외시킬 수 있습니다. 제외된 부위는 빨간색으로 표시됩니다.



휴머노이드 아바타 마스크 설정



제네릭 아바타 마스크 설정

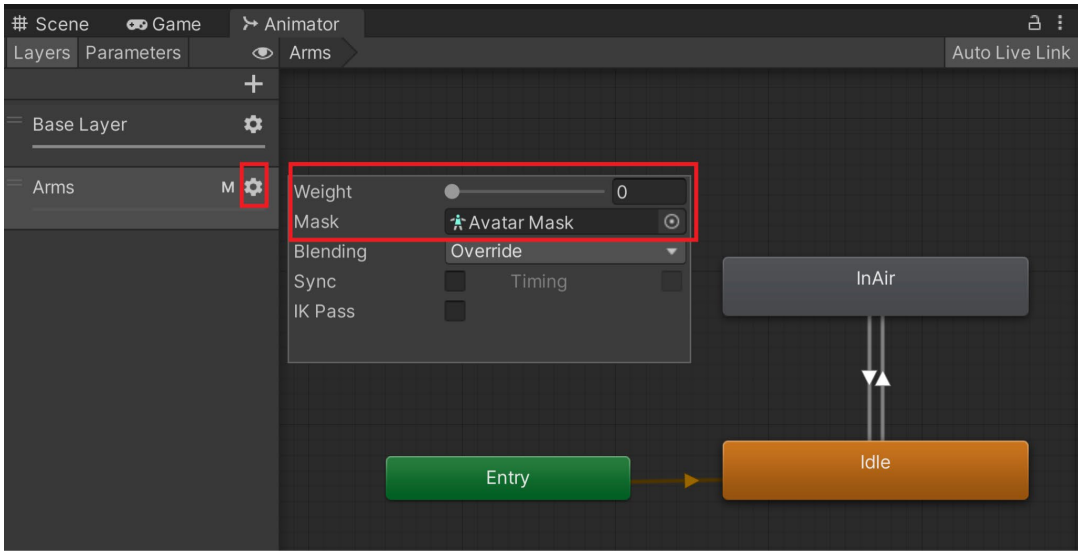
Transform 섹션에서는 캐릭터의 골격 구조를 트리 형태로 확인할 수 있으며, 제네릭 애니메이션 유형일 때 사용합니다. 캐릭터의 아바타를 추가하고 해당 아바타에서 골격을 로드해야 합니다.

트리를 펼쳐 모든 뼈대를 표시하고 토글 버튼을 통해 켜고 끌 수 있습니다. 기본 애니메이션을 오버라이드하려면 항목을 선택합니다. 선택하지 않은 항목은 제외되므로 선택하지 않은 뼈대에는 기본 애니메이션이 적용됩니다.

예시에서는 캐릭터의 팔만 선택했으므로 팔에서는 새로운 애니메이션이 재생되고 캐릭터의 나머지 부위에서는 기본 애니메이션이 재생됩니다.

애니메이터에 추가한 새로운 레이어가 점프 애니메이션을 사용하며 이때 팔이 들어 올려집니다. 전환은 Base Layer와 동일합니다. 톱니바퀴 설정 아이콘을 클릭하여 Mask 슬롯에 아바타 마스크를 적용할 수 있습니다.

가중치를 조정하여 이 레이어를 제어할 수 있습니다. 가중치가 0이면 기본 레이어는 영향을 받지 않습니다. 가중치가 1이면 기본 레이어의 뼈대를 완전히 오버라이드합니다. 가중치를 조절하여 얻은 다양한 결과를 통해 애니메이션 간에 부드러운 블렌딩을 구현할 수 있습니다.



새 레이어를 추가하고 아바타 마스크로 제어하기

C# 스크립트에서 레이어 가중치를 제어하려면 괄호 안의 파라미터**(레이어 인덱스, 레이어 가중치)**를 조정합니다. 아래 예시 코드에서 Arms 레이어의 인덱스는 1이며 Shift 키를 누르면 가중치가 1이고 Shift 키를 해제하면 가중치가 0입니다.

```
if (Input.GetKeyDown(KeyCode.LeftShift))
{
    animator.SetLayerWeight(1, 1);
}
if (Input.GetKeyUp(KeyCode.LeftShift))
{
    animator.SetLayerWeight(1, 0);
}
```

애니메이션 레이어 가중치를 제어하는 코드

플레이 모드에서 Shift 키를 누른 상태로 걸으면 스크립트에서 두 애니메이션을 결합합니다.

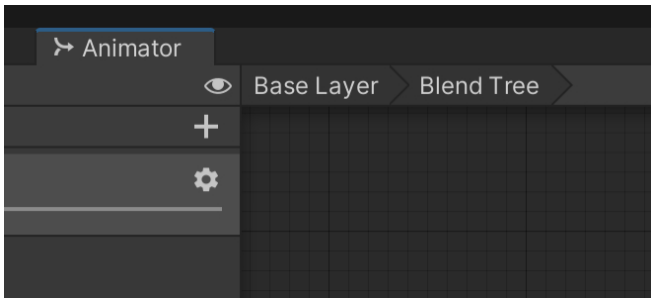
아바타 마스크를 사용하여 결합된 애니메이션

블렌드 트리

입력 파라미터에 따라 세 가지 이상의 애니메이션을 동적으로 함께 블렌딩해야 하는 경우 블렌드 트리를 사용합니다. 유니티에서 제작한 Starter Assets – ThirdPerson 패키지에서는 블렌드 트리를 사용하여 3인칭 캐릭터 모션을 제어합니다.

Animator 컨트롤러 창을 오른쪽 클릭하고 컨텍스트 메뉴에서 **Create state > From New Blend Tree**를 선택하여 블렌드 트리를 추가합니다.

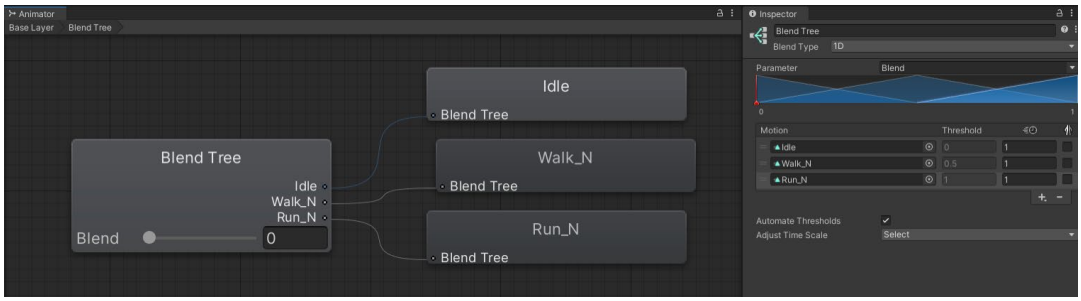
더블 클릭하여 블렌드 트리 설정에 진입합니다. 블렌드 트리에서 나가려면 좌측 상단의 버튼을 사용합니다.



기본 레이어를 클릭하여 블렌드 트리를 종료합니다.

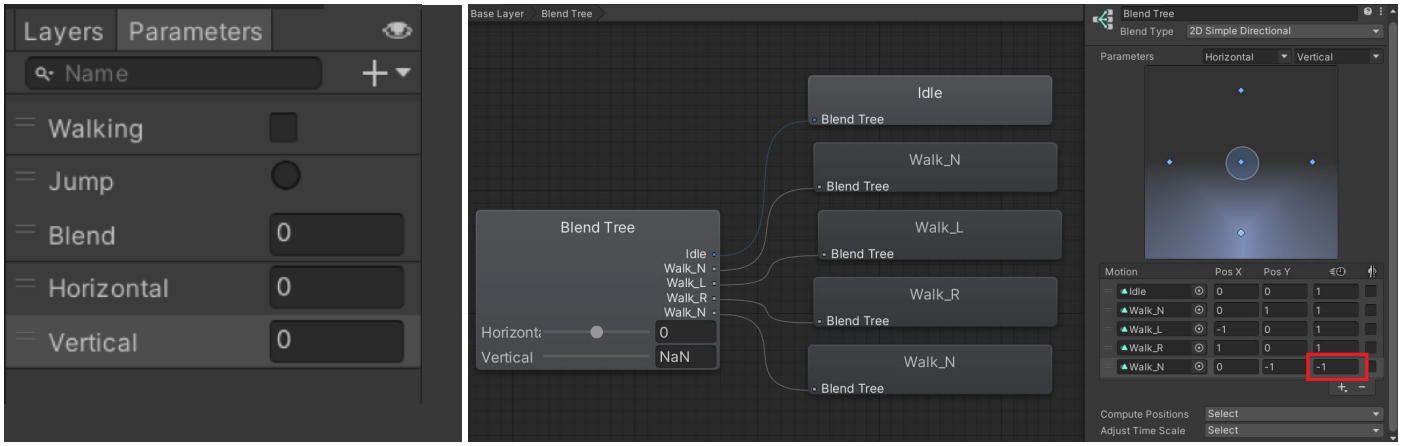
블렌드 트리 노드를 클릭하여 함께 블렌딩할 애니메이션을 추가합니다.

아래 예시에서는 대기 상태, 걷기, 달리기의 3가지 애니메이션을 블렌딩합니다. W 키 또는 위쪽 화살표 키를 누르는 것에 따라 0.0에서 1.0으로 변하는 플로트 입력을 바탕으로 애니메이션이 블렌딩됩니다.



블렌드 트리 설정

블렌드 트리는 단일 입력(1D 블렌드) 또는 두 가지 입력(2D 블렌드)을 기반으로 설정할 수 있습니다. 예를 들어 WASD 키를 눌러 캐릭터의 종횡 이동을 제어합니다.



간단한 2D 블렌드 트리

2D 블렌드 트리에는 다양한 방향을 처리할 수 있도록 2개의 파라미터가 필요합니다. Pos X(가로) 및 Pos Y(세로)의 최댓값을 설정할 수 있습니다.

위 이미지에서 뒤로 걷기 애니메이션은 Walk_N 애니메이션을 재사용하며 재생 속도를 -1로 설정해 역재생하여 구현한다는 것을 볼 수 있습니다.

이제 코드에서 W 및 S 키 또는 위쪽 및 아래쪽 화살표 키의 입력 축(Input GetAxis("Vertical"))을 애니메이터의 플로트 파라미터에 전달할 수 있습니다. 이는 1D 블렌드 셰이프의 블렌딩을 제어합니다.

```
Unity Message | 0 references
void Update()
{
    animator.SetFloat("Blend", Input.GetAxis("Vertical"));
```

코드를 통한 블렌드 트리 제어

2D 블렌드는 파라미터가 두 개이므로 두 가지 입력이 필요합니다.

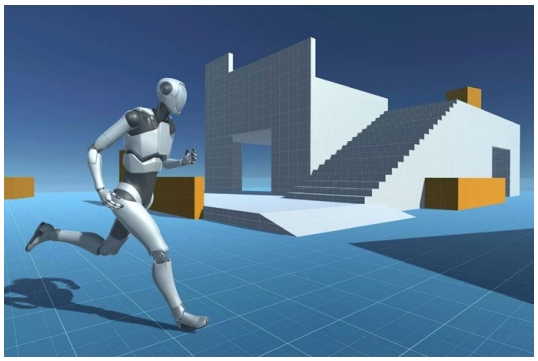
```
Unity Message | 0 references
void Update()
{
    animator.SetFloat("Blend", Input.GetAxis("Vertical"));
```

2D 블렌드 코드

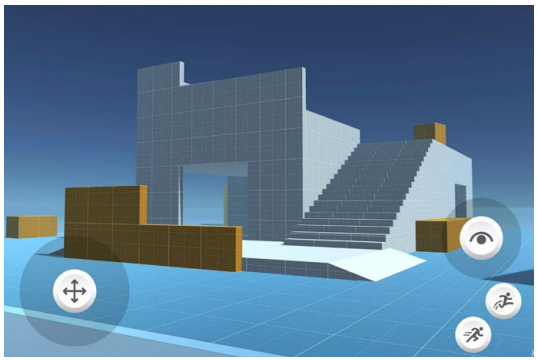
캐릭터 컨트롤러

Character Controller 컴포넌트는 플레이어블 캐릭터를 제어하기 위해 사용되며, 슬로프 경사 제한이나 계단 높이 제한 같은 프로퍼티를 설정할 수 있습니다. Character Controller 컴포넌트는 주로 리지드바디 물리를 사용하지 않는 3인칭 또는 1인칭 플레이어 컨트롤에 사용됩니다.

Starter Assets – ThirdPerson 패키지뿐만 아니라 유니티에서 제작한 Starter Assets – FirstPerson 패키지도 Unity 에셋 스토어에서 다운로드할 수 있습니다. 두 패키지 모두 이미 설정된 캐릭터 컨트롤러를 제공하므로 씬을 빠르게 프로토타이핑할 수 있습니다.



Starter Assets – ThirdPerson



Starter Assets – FirstPerson

플레이어블 캐릭터의 캐릭터 컨트롤러를 설정하는 주요 단계를 살펴보겠습니다.

1. 3가지 애니메이션 클립으로 블렌드 트리를 생성합니다. 전방으로 걷기 또는 달리기, 대기 상태, 후방으로 걷기 또는 달리기 클립입니다. **Automate Thresholds** 체크박스를 선택 해제합니다. 후방으로 걷기의 임곗값을 -1, 대기 상태를 0, 전방으로 걷기를 1로 설정합니다.
2. 결과는 다음과 같을 것입니다.

    a. W 키를 누르면 세로 축이 1로 설정되고 캐릭터가 전방으로 걸어갑니다.

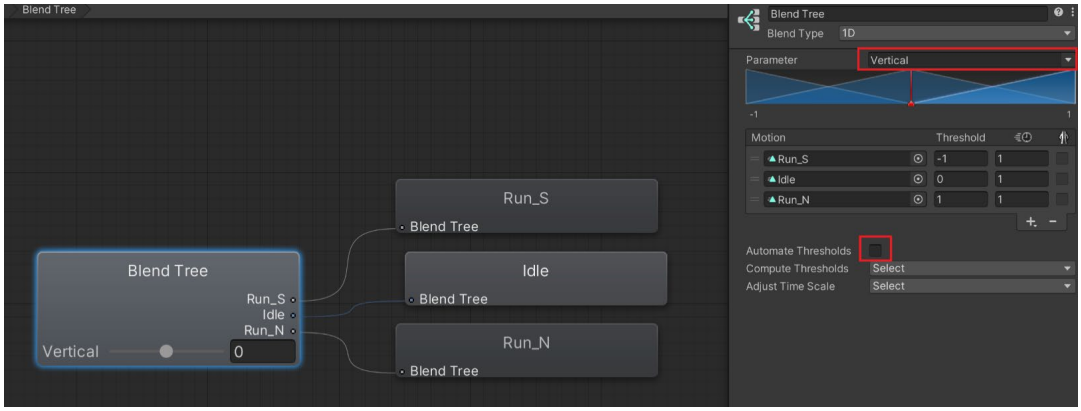
    b. S 키를 누르면 세로 축이 -1로 설정되고 캐릭터가 후방으로 걸어갑니다.

    c. A 키를 누르면 가로 축이 -1로 설정되고 캐릭터가 왼쪽으로 걸어갑니다.

    d. D 키를 누르면 가로 축이 1로 설정되고 캐릭터가 오른쪽으로 걸어갑니다.

    이제 블렌드 트리에서 세로 및 가로 축에 올바른 모션을 매칭합니다.

3. 모션을 제어하는 플로트 파라미터 Vertical을 생성하고 아래 이미지와 같이 1D 블렌드의 파라미터로 설정합니다.



플레이어블 캐릭터의 블렌드 트리 애니메이션

간단한 캐릭터 컨트롤러 조작으로 캐릭터 이동을 제어하는 스크립트가 필요합니다.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

Unity Script (2 asset references) | 0 references
public class BasicMove : MonoBehaviour
{
    private Animator animator;
    private CharacterController characterController;
    public float walkSpeed = 2.5f;
    public float rotateSpeed = 1.0f;

    Unity Message | 0 references
    void Start()
    {
        animator = GetComponent<Animator>();
        characterController = GetComponent<CharacterController>();
    }

    Unity Message | 0 references
    void Update()
    {
        //Rotation
        transform.Rotate(0, Input.GetAxis("Horizontal") * rotateSpeed, 0);
        //movement
        Vector3 forward = transform.TransformDirection(Vector3.forward);
        float curSpeed = walkSpeed * Input.GetAxis("Vertical");
        characterController.SimpleMove(forward * curSpeed);
        //Animation
        animator.SetFloat("Vertical", Input.GetAxis("Vertical"));
    }
}
```

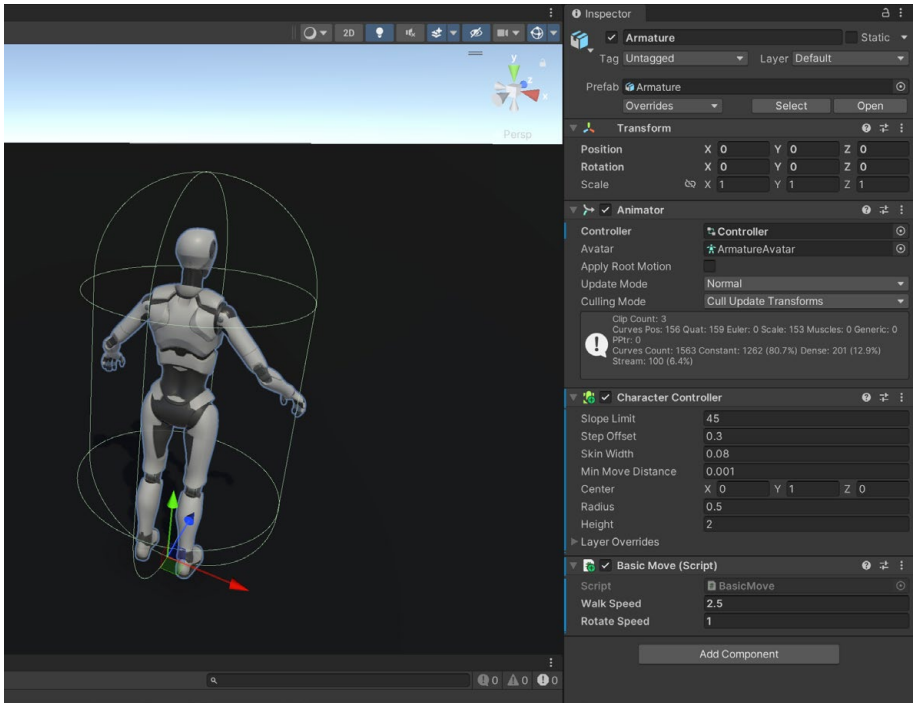
플레이어의 이동과 회전을 제어하는 기본 이동 스크립트

4. 캐릭터에 스크립트와 Character Controller 컴포넌트를 추가합니다.

5. Step Offset을 조정하여 캐릭터가 계단을 오를 수 있도록 설정합니다. 캐릭터가 아주 높은 계단을 오를 수 있어야 한다면 이 숫자를 높게 설정합니다. 하지만 너무 높으면 캐릭터가 작은 벽도 넘어가 버릴 수 있습니다.

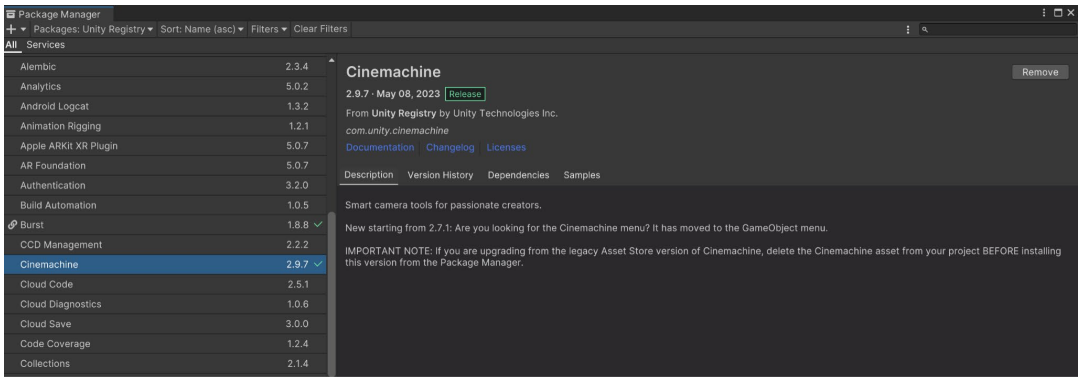
6. Slope Limit 프로퍼티를 조정하여 캐릭터가 경사면을 걸어 오를 수 있도록 설정합니다. 기본값은 45도지만 캐릭터가 더 가파른 표면을 오를 수 있도록 더 높게 설정할 수도 있습니다. 90도 이상으로 설정하면 캐릭터가 건물 벽을 걸어 올라갈 수도 있습니다.

7. 캐릭터 컨트롤러의 위치를 올려 캐릭터 중심과 일치시키려면 Center Y 값을 1로 설정합니다.
8. Height는 2로 설정하여 캐릭터의 키에 맞춥니다. 캐릭터의 키가 더 크다면 Height 값을 더 높게 설정합니다.
9. Radius 값을 변경하여 너비를 조정합니다. 캐릭터의 폭이 넓으면 1 이상, 좁으면 0.3 이하로 설정합니다.



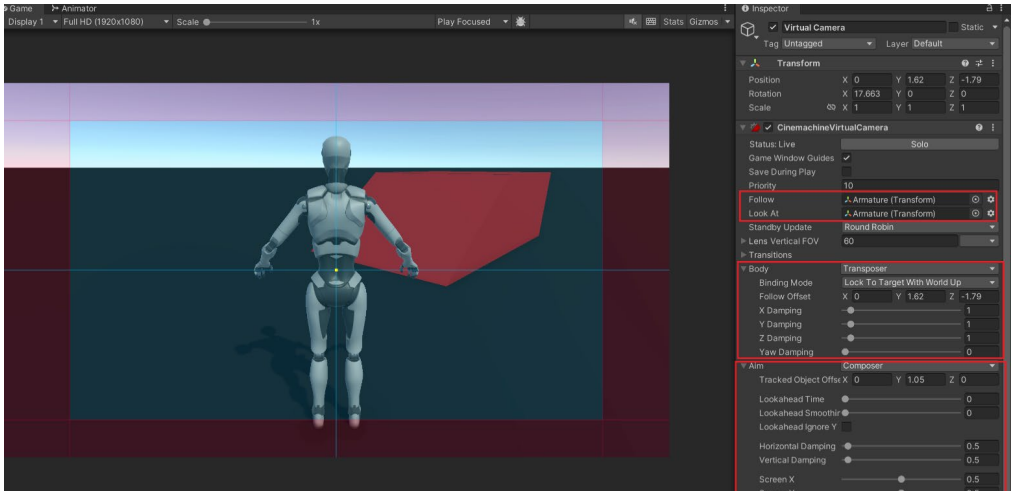
플레이어 캐릭터에 추가된 Character Controller 컴포넌트

Cinemachine은 고급 카메라 컨트롤, 절차적 애니메이션, 스마트 카메라 동작을 통해 게임 및 시뮬레이션을 위한 동적 카메라 시스템 제작 프로세스를 간소화하는 Unity 에셋입니다. Cinemachine에 대해서는 이 책의 후반부에서 자세히 살펴보겠습니다. 지금은 **Window > Package Manager**의 Unity Registry 메뉴에서 Cinemachine을 설치합니다.



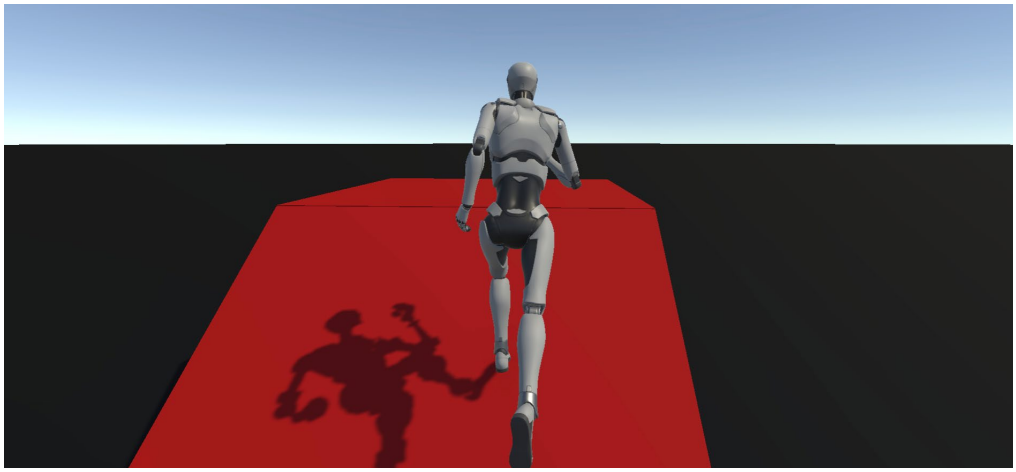
패키지 관리자에서 Cinemachine 패키지 선택

GameObject > Cinemachine > Virtual Camera를 선택하여 플레이어를 따라다니는 카메라를 추가합니다. **Follow**와 **Look At** 필드에 캐릭터를 추가하고 값을 조정하여 씬 뷰에서 카메라 뷰를 설정합니다.

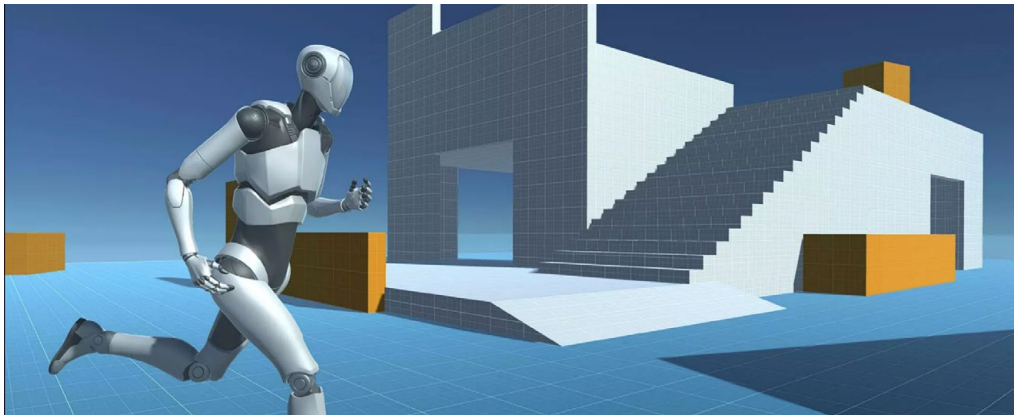


팔로우 카메라의 Cinemachine 가상 카메라 설정

이제 게임을 플레이하면 WASD 또는 화살표 키로 캐릭터를 제어할 수 있습니다.



캐릭터 컨트롤러를 적용한 플레이어블 캐릭터



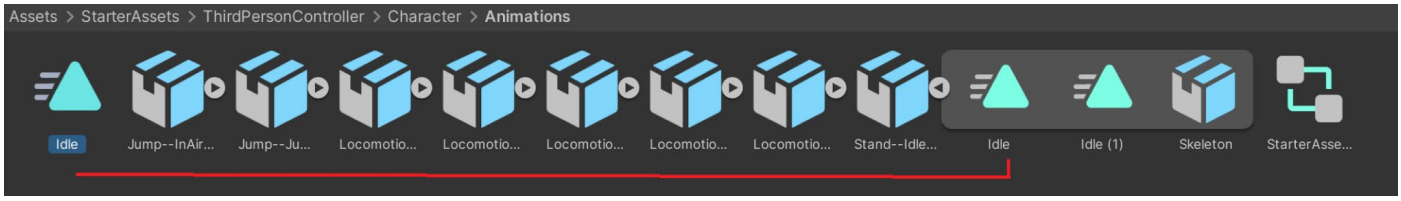
3인칭 Starter Assets 팩

Unity에서 애니메이션 제작하기

모든 오브젝트를 Unity에서 직접 애니메이션화할 수 있습니다. **Window > Animation > Animation**을 선택하여 Animation 뷰를 엽니다.

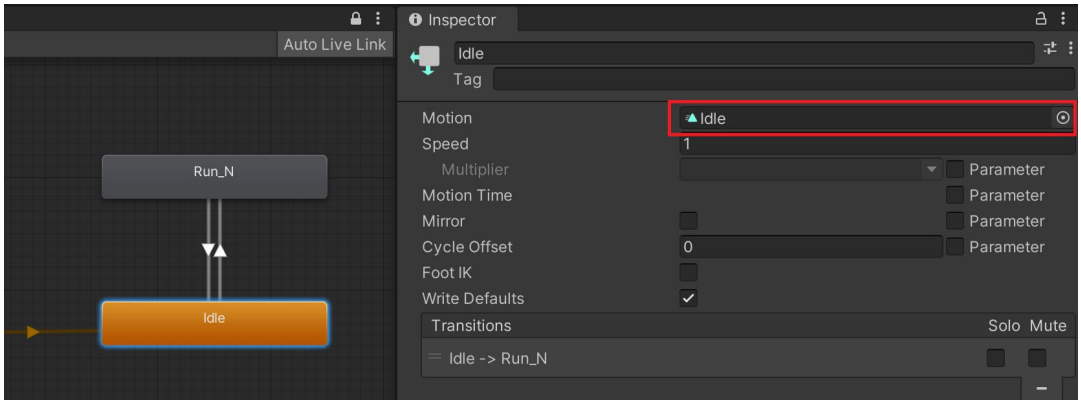
FBX 파일에 내장된 애니메이션은 읽기 전용입니다. 클립을 편집하려면 복제해야 하지만, 클립을 복제하면 더 이상 애니메이션 유형을 변경할 수 없으니 주의하세요. 애니메이션 유형을 변경하려면 FBX 파일의 원본 클립에 액세스해야 합니다.

애니메이션을 선택하고 **Ctrl+D** 또는 **Edit > Duplicate**를 눌러서 복제합니다.



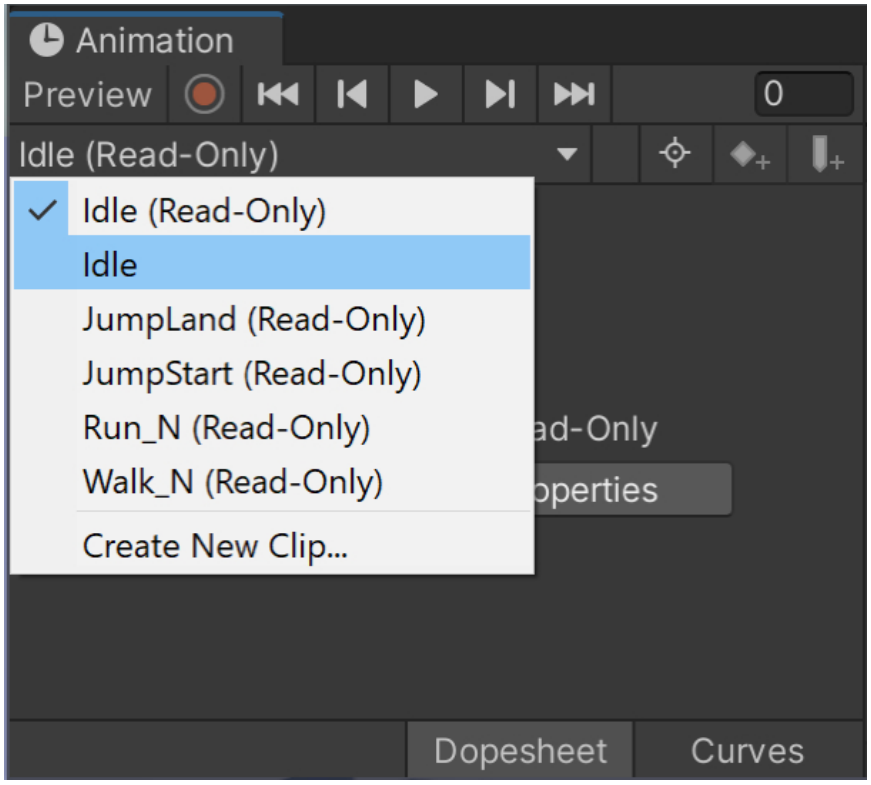
애니메이션 복제

복제한 애니메이션을 편집하려면 캐릭터의 Animator 컨트롤러에 추가합니다. Idle 노드를 클릭하고 **Motion** 필드에 새 애니메이션 파일을 드래그합니다.



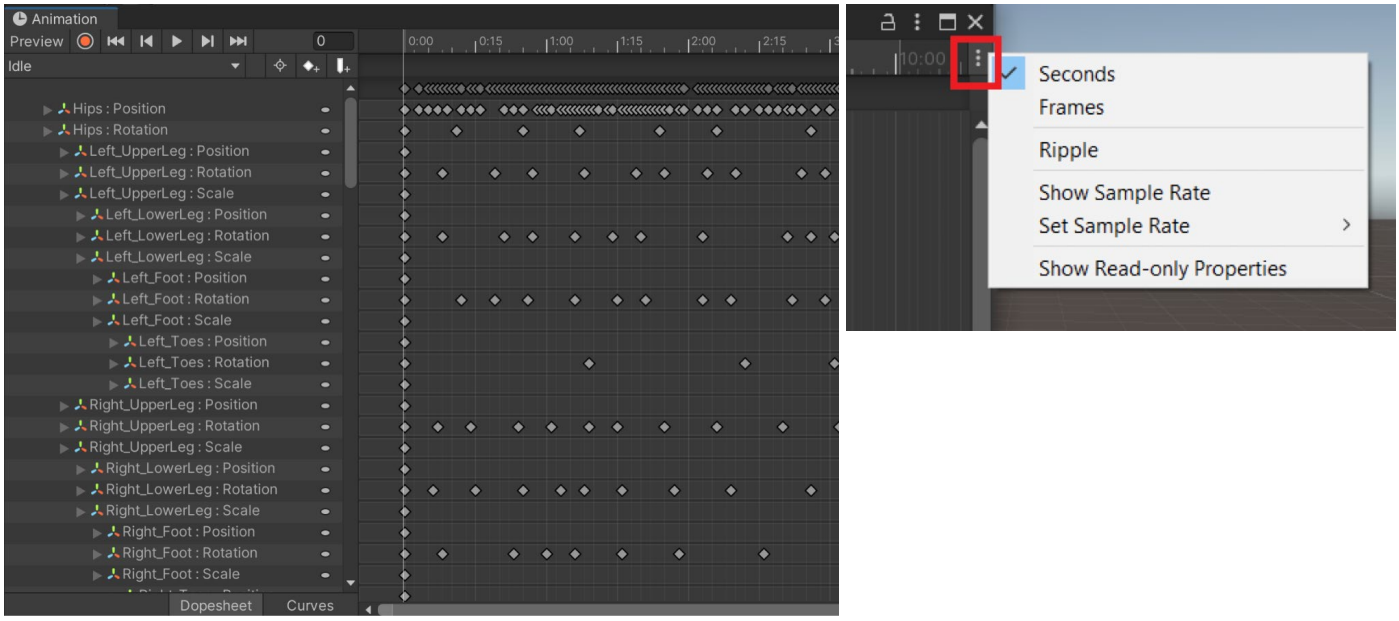
새 클립을 Motion 슬롯으로 드래그하여 애니메이션 클립을 교체합니다.

캐릭터를 선택하면 Animation 뷰에서 직접 Animator 컨트롤러의 애니메이션에 액세스할 수 있습니다. 좌측 패널의 드롭다운 메뉴에서 선택할 수 있습니다. 복제된 클립은 편집할 수 있지만 내장된 클립은 읽기 전용입니다.



임베디드 애니메이션을 수정하려면 이를 복제해야 합니다.

이 드롭다운 메뉴에서 **Create New Clip**을 선택하여 새 애니메이션 클립을 생성할 수도 있습니다.

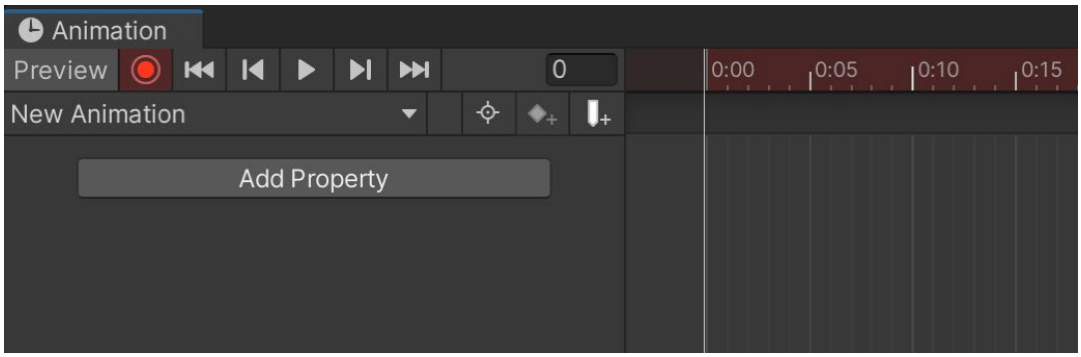


키프레임 보기(왼쪽 상단) 및 애니메이션 옵션 변경(오른쪽 상단)

키프레임을 클릭하여 선택합니다. 키프레임을 삭제, 이동, 복제, 편집할 수 있습니다. 편집하려면 녹화 버튼을 켭니다.

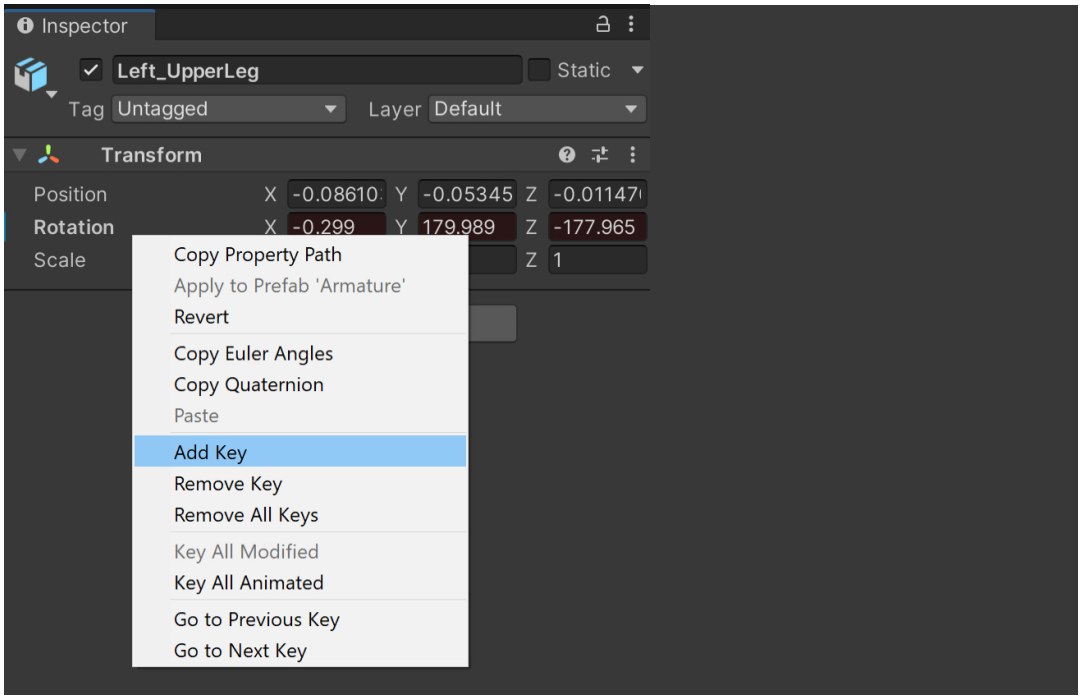
Animation 창 우측 상단의 점 3개 아이콘을 클릭하여 옵션을 변경할 수 있습니다. 타임라인을 초 또는 프레임 단위로 볼 수 있고, 샘플 레이트를 30fps 또는 60fps로 변경할 수 있습니다.

새 클립에서 순운동학(FK) 애니메이션을 제작합니다. 새로운 클립을 생성하면 키프레임이 설정되지 않아 비어 있습니다. 녹화 버튼을 눌러 키프레임 설정을 시작합니다.



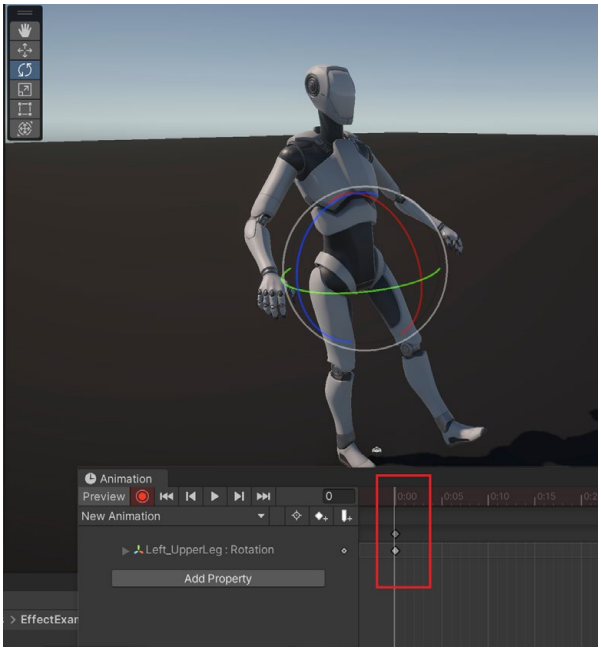
새 클립에는 키프레임이 없습니다. 녹화 버튼을 눌러 키프레임 생성을 시작합니다.

키프레임을 설정하려면 계층 구조에서 뼈대를 선택하고 인스펙터에서 Rotation을 오른쪽 클릭하여 **Add Key**를 선택합니다. 이렇게 하면 초기 키프레임이 설정됩니다.



Rotation 필드 옆을 오른쪽 클릭하여 뼈대에 키를 추가합니다.

뼈대는 키프레임 간에 애니메이션화되므로 시간이 지나도 뼈대를 한 위치에 고정하려면 이를 유용하게 사용할 수 있습니다.



뼈대를 회전하여 키프레임을 설정합니다.

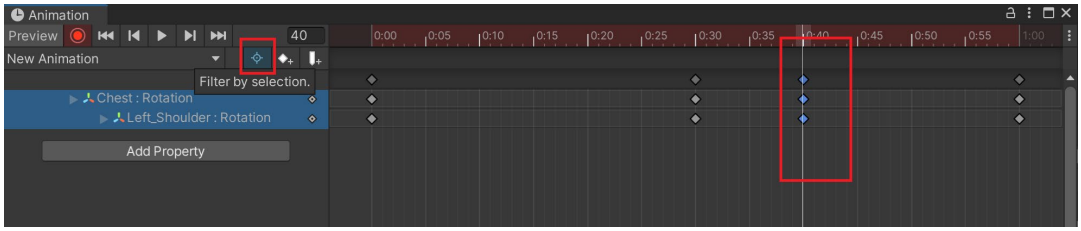
예를 들어 다리를 2초 동안 정지 상태로 유지하고 싶다면 0초에 키를 추가하고 2초로 이동한 다음 키를 하나 더 추가합니다. 이렇게 하면 다리가 정지 상태로 유지됩니다. 해당 시점이 지난 다음에 회전을 시작할 수 있습니다.

씬 뷰에서 회전 툴을 사용하여 뼈대를 회전하고 새 키프레임을 추가합니다.

Add keyframe 버튼을 사용하여 애니메이션의 모든 뼈대에서 선택된 프레임에 키프레임을 추가해도 됩니다.

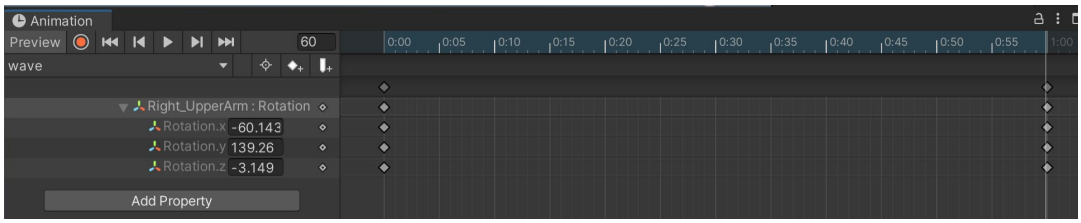
Add keyframe 옵션을 사용하여 특정 프레임의 모든 뼈대에 새 키프레임을 추가합니다.

특정 뼈대에만 키를 설정하려면 목록에서 뼈대를 선택하고 **Filter by selection** 버튼을 클릭합니다. 그런 다음 Add keyframe 버튼을 누르면 해당 뼈대에만 키가 추가됩니다. 이 방법이 각 뼈대의 Rotation을 오른쪽 클릭한 다음 키를 추가하는 것보다 빠릅니다.



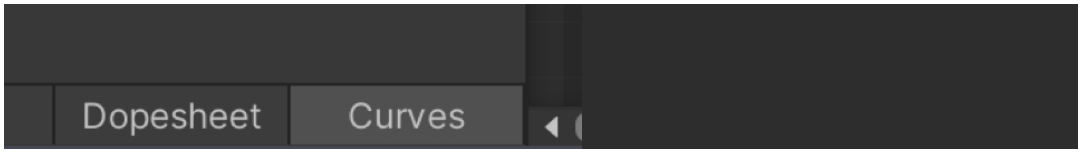
선택한 항목으로 필터링하여 선택한 뼈대에만 키를 추가합니다.

회전, 트랜스폼, 스케일의 X, Y, Z 축 값을 개별적으로 조정하거나 특정 값을 입력할 수 있습니다.



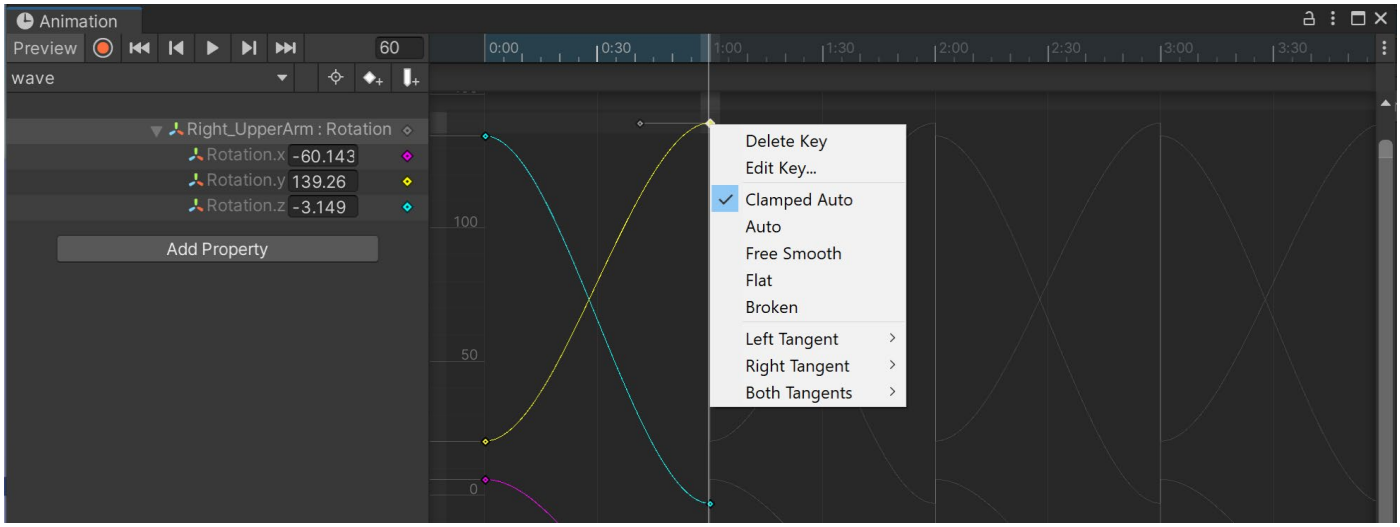
X, Y, Z축 키프레임 설정

Animation 창 하단에서 키프레임을 표시하는 **Dopesheet** 뷰를 Curves 뷰로 전환하고 개별 키프레임 커브를 클릭하여 편집하거나 이동할 수 있습니다.



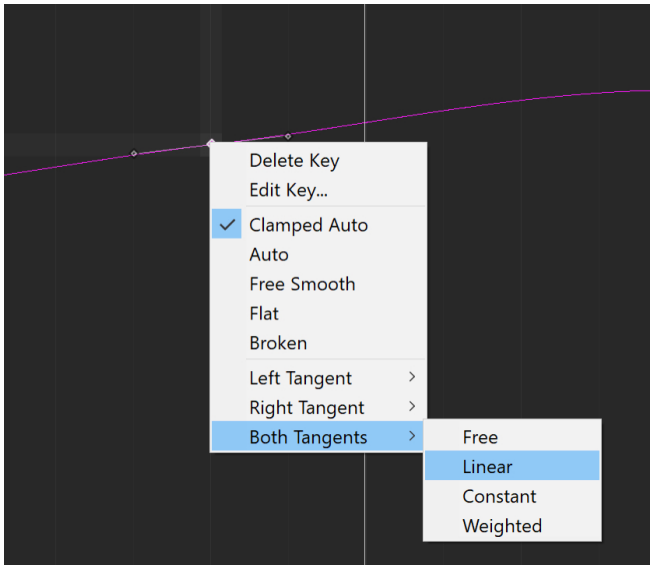
하단의 버튼을 사용하여 Dopesheet(키프레임)와 Curves 사이를 변경합니다.

오른쪽 클릭으로 커브의 모양을 변경하여 시작 및 종료 위치 사이의 이징(easing)을 조정할 수 있으며 베지어 핸들을 사용해 커브를 설정할 수도 있습니다.



커브 편집

커브는 기본적으로 **Clamped Auto**로 설정되며 이 설정에서는 키프레임 간 모션에 가속(ease in) 및 감속(ease out)이 적용됩니다. 느리게 시작하여 중간에는 속도가 빨라지고 끝으로 갈수록 다시 느려져 부드러운 모션이 구현됩니다.

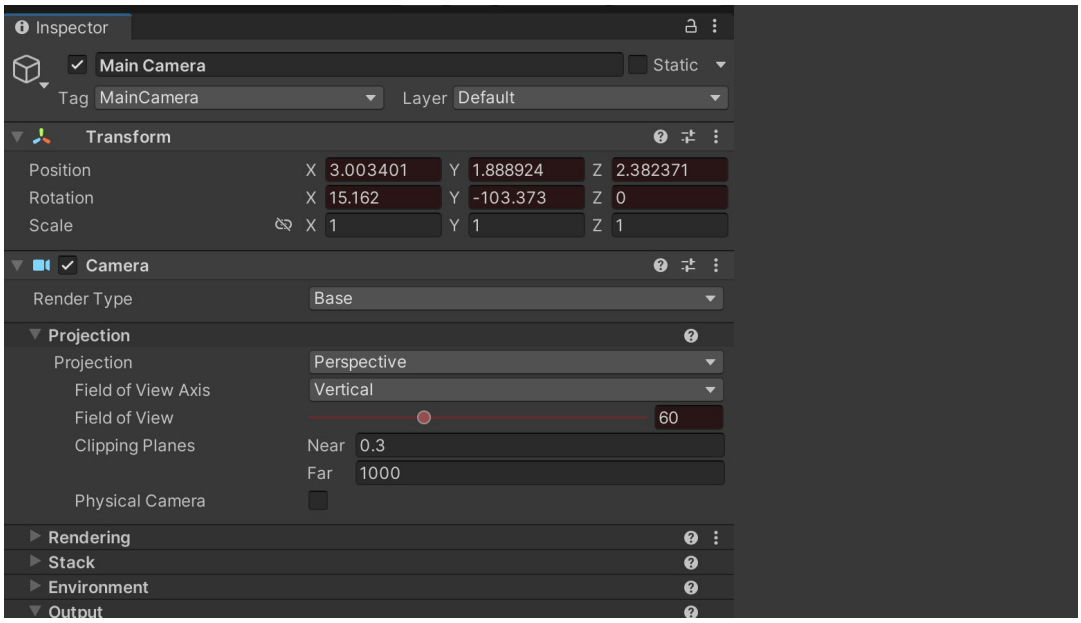


Linear 탄젠트를 선택하여 키프레임에서 커브 제거

키프레임 간 이징을 없애기 위해 모션에서 커브를 제거하려면 키프레임을 오른쪽 클릭하고 Both Tangents를 **Linear**로 설정합니다. 왼쪽 또는 오른쪽에서 선형으로만 재생되도록 커스터마이즈할 수도 있습니다.

오브젝트와 그 컴포넌트를 애니메이션화하고, 항목을 켜고 끄며, 머티리얼 값이나 광원의 강도를 수정하고, 애니메이션을 사용하여 카메라 시야각(FOV)의 위치를 지정할 수 있습니다.

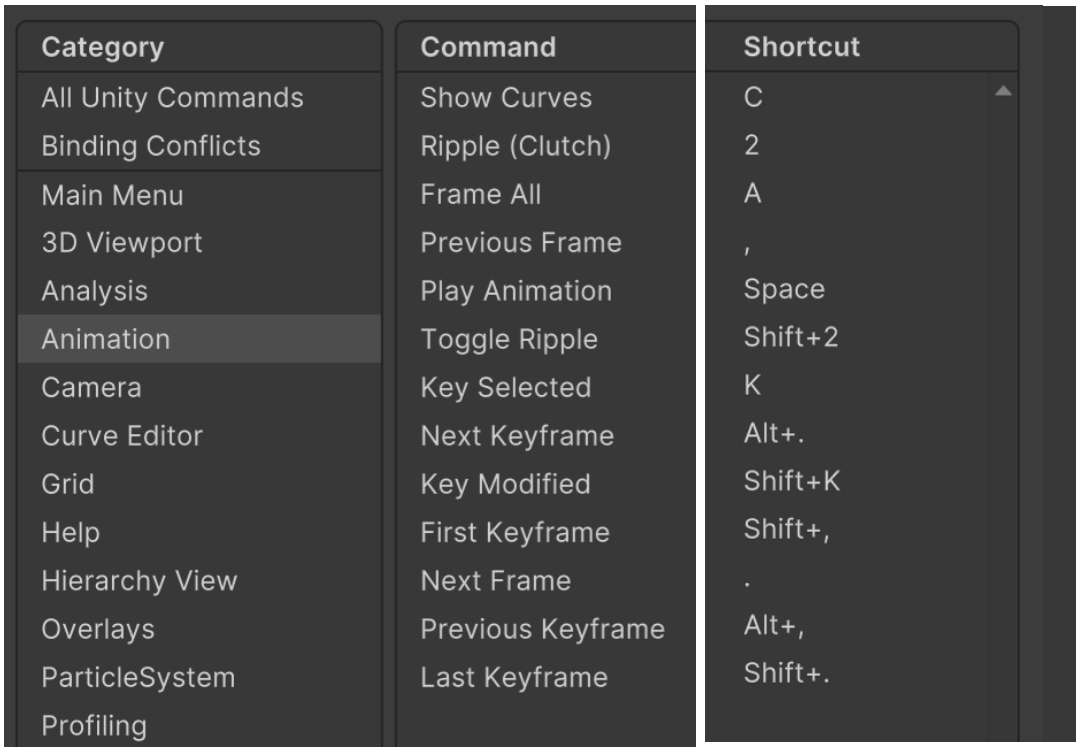
애니메이션이 적용된 설정은 인스펙터에서 빨간색으로 표시되고 녹화 모드를 끄면 파란색으로 표시됩니다.



애니메이션이 적용된 항목은 이 이미지의 Position 및 Rotation 값처럼 빨간색 또는 파란색으로 변경됩니다.

Animation 창에서 유용한 단축키

Animation 창의 단축키를 표시하려면 **Edit > Shortcuts**로 이동합니다. 목록에서 **Animation**을 선택합니다.



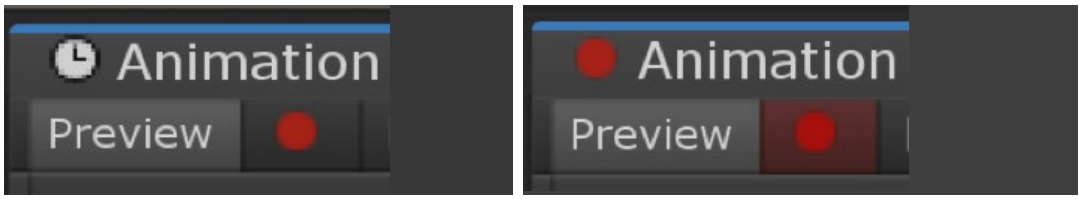
Windows에서 유용한 애니메이션 단축키

Windows용 Animation 창 줌 컨트롤

선택한 키/커브 프레이밍	F
모든 커브 프레이밍	A
줌(zoom) 인/아웃	마우스 휠 (또는) Alt+오른쪽 클릭한 상태로 드래그
세로 줌	Alt+Shift+오른쪽 클릭한 상태로 드래그

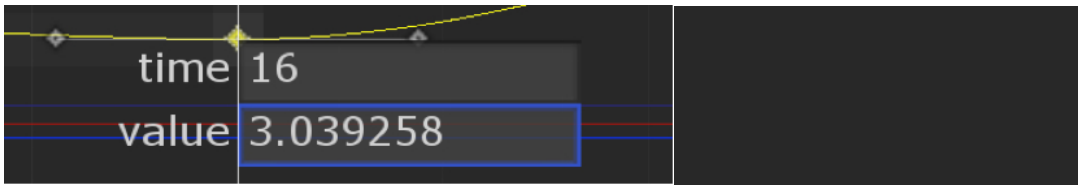
Windows에서 Animation 창을 탐색하는 단축키

Preview 모드와 녹화 모드



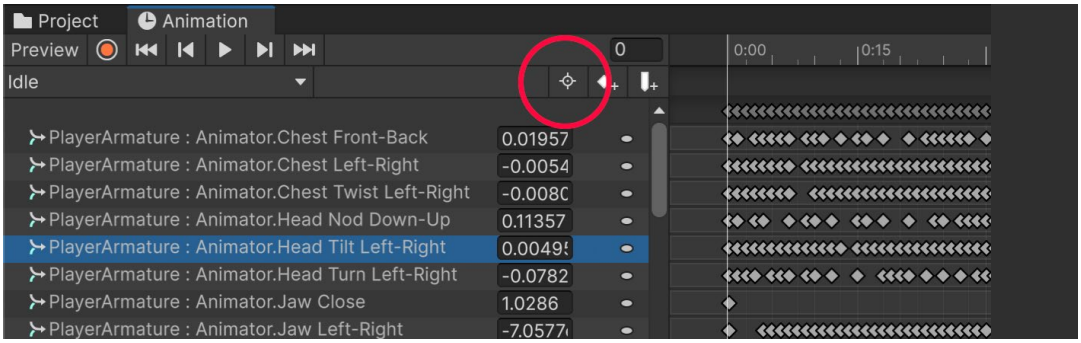
Preview 모드에서는 직접 키프레임을 설정해야 하는 프로퍼티에 후보 키를 설정할 수 있습니다. 녹화 모드에서는 모든 수정 사항이 자동으로 클립에 추가됩니다.

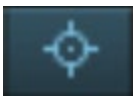
키프레임에 정확한 값 설정



커브 뷰에서 키프레임을 하나 이상 선택한 다음 Enter 키를 누릅니다. 정확한 값과 시간을 입력할 수 있는 팝업 창이 표시됩니다. 탄젠트는 수정되지 않습니다.

선택한 항목으로 필터링



이 아이콘은 **선택한 항목으로 필터링**하는 버튼입니다. 

활성화하면 선택한 오브젝트의 프로퍼티만 씬 뷰에 표시됩니다. 비활성화하면 애니메이션 클립의 모든 프로퍼티가 표시됩니다.

애니메이션 이벤트

이벤트는 C# 스크립트에서 함수 실행을 트리거하는 애니메이션에서 호출될 수 있습니다. 이러한 함수는 게임 오브젝트에 연결된 모든 스크립트에 있을 수 있습니다.

아래 예시에는 왼발이나 오른발이 지면과 닿을 때 물이 튀게 하고 발소리를 재생하는 두 개의 커스텀 함수가 있습니다.

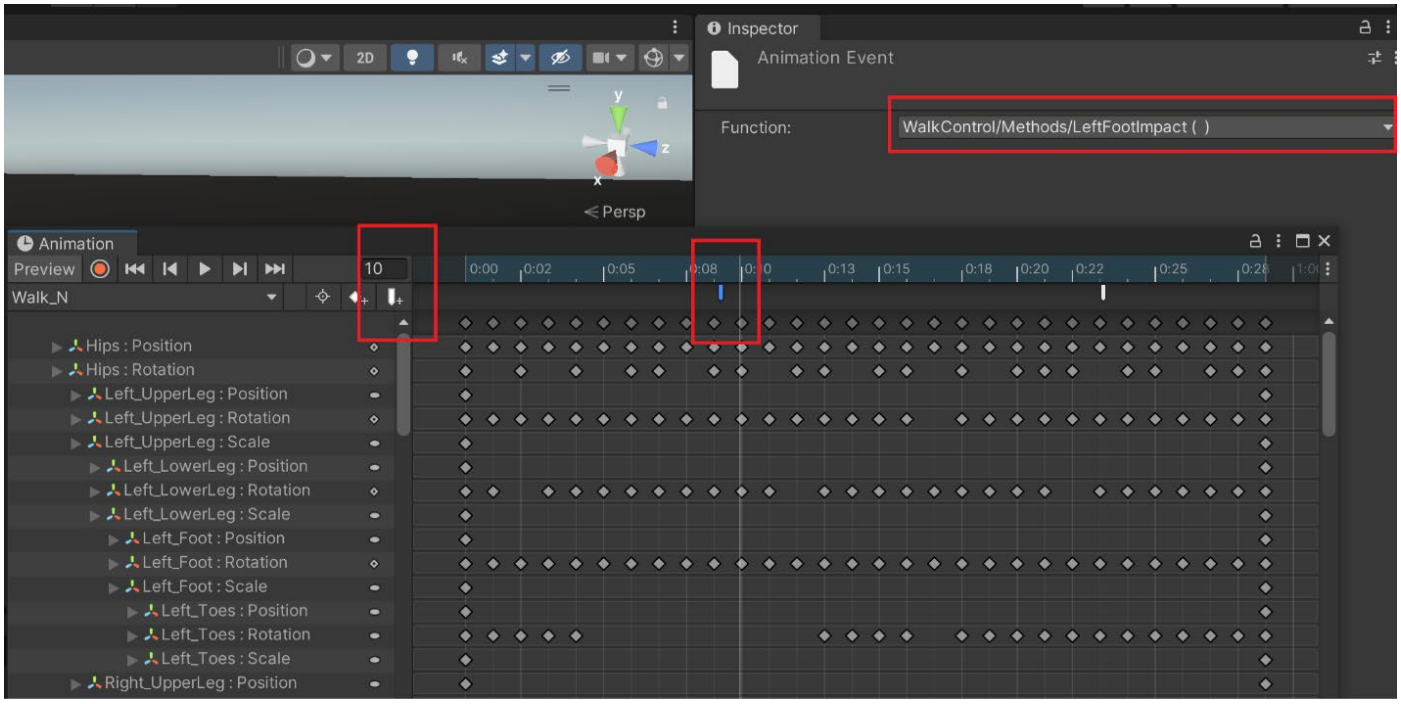
```
Unity Script (1 asset reference) | 0 references
public class WalkControl : MonoBehaviour
{
    private Animator animator;
    private AnimatorStateInfo info;
    public GameObject splash;
    public Transform spawnPoint1, spawnPoint2;
    private AudioSource audioPlayer;

    Unity Message | 0 references
    private void Start()
    {
        animator = GetComponent<Animator>();
        audioPlayer = GetComponent<AudioSource>();
    }

    0 references
    public void LeftFootImpact()
    {
        Instantiate(splash, spawnPoint1.position, spawnPoint1.rotation);
        audioPlayer.Play();
    }
    0 references
    public void RightFootImpact()
    {
        Instantiate(splash, spawnPoint2.position, spawnPoint2.rotation);
        audioPlayer.Play();
    }
```

애니메이션 이벤트에 의해 트리거되는 커스텀 함수

Animation 창에서 애니메이션의 올바른 시점에 이벤트 시간을 설정할 수 있습니다. 이벤트 아이콘을 클릭하여 이벤트를 추가합니다.

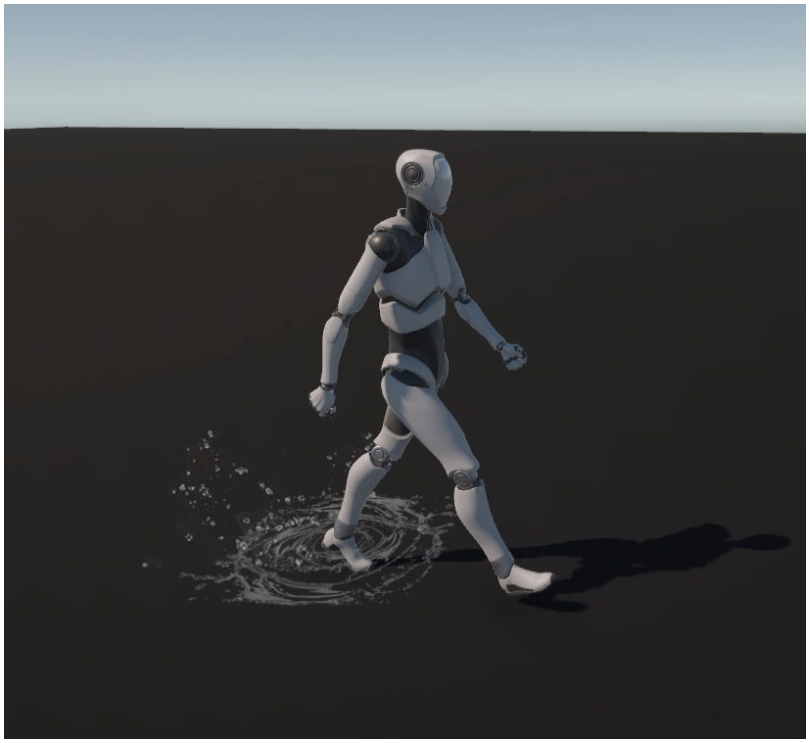
새 이벤트를 추가하는 이벤트 아이콘

스크립트에서 트리거할 함수를 선택하라는 프롬프트가 인스펙터에 표시됩니다.



애니메이션에 이벤트 추가

애니메이션을 재생하면 발이 웅덩이에 닿을 때 물이 튀는 효과가 생성되고 걷는 주기에 맞춰 발소리가 재생될 것입니다.



애니메이션 도중 이벤트를 발생시키는 코드

고급 애니메이션 기능

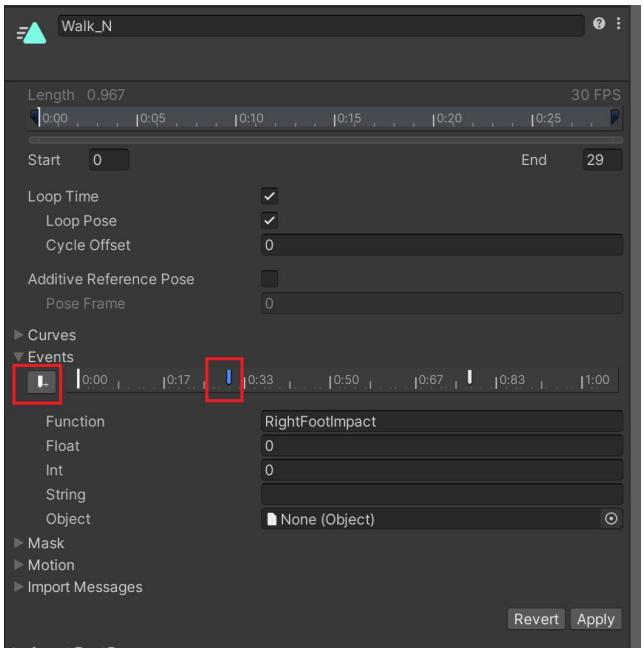
읽기 전용 애니메이션의 이벤트

FBX 파일에 내장된 읽기 전용 클립에도 이벤트를 추가할 수 있습니다. FBX 파일에서 복제한 클립이 아닌 경우 Animation 창에서 편집할 수 없습니다.

FBX 파일의 애니메이션 섹션에서 Events 탭을 펼칩니다. 화면 하단의 미리 보기 창을 보면서 클립의 타이밍을 조정할 수 있습니다. **Add Event**를 클릭합니다.

새 이벤트를 추가하는 이벤트 아이콘

이를 위해서는 스크립트에서 함수 이름을 직접 입력해야 합니다. 정수, 플로트, 문자열 같은 필수 파라미터를 추가할 수 있는 옵션도 있습니다.

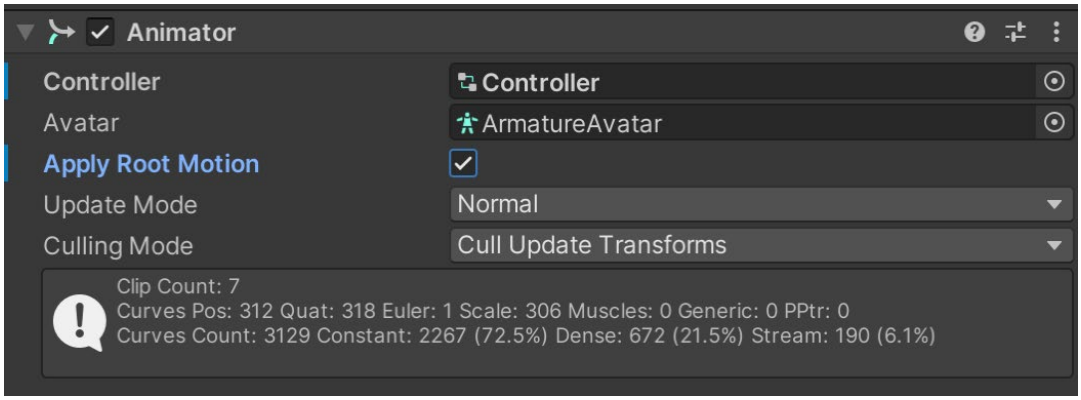


읽기 전용 클립에 이벤트 추가

루트 모션

많은 클립은 고정 애니메이션으로 만들어지며, 걷기 애니메이션의 경우 걷기가 재생되는 동안 캐릭터의 위치는 한곳에 고정됩니다. 스크립트를 사용하여 캐릭터의 이동을 제어할 수 있습니다.

일부 클립에는 캐릭터를 3D 공간에서 움직이는 이동 키프레임이 포함되어 있습니다. 예를 들어 걷기 애니메이션에서 캐릭터는 클립이 재생되는 동안 전방으로 움직이게 됩니다. 하지만 클립이 처음으로 돌아와 반복되면 캐릭터도 원점으로 다시 돌아오며 어색한 동작이 연출됩니다.



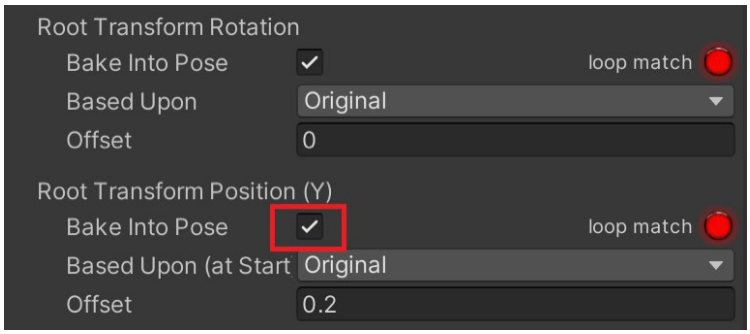
Animator 컴포넌트의 Apply Root Motion

Animator 컴포넌트에는 루트 모션을 적용하는 옵션이 있습니다. 이 박스를 선택하면 클립이 반복되면서 원점으로 돌아가는 대신 현재 위치에서 계속 재생됩니다. 따라서 캐릭터가 실감 나게 전방으로 걸어가게 됩니다.

루트 모션을 사용하면 캐릭터의 키프레임 타이밍에 맞춰 움직임이 이루어지기 때문입니다. 발이 땅에 닿으며 캐릭터가 실감 나게 전방으로 움직이며, 코드를 사용하여 캐릭터를 움직였을 때 발생할 수 있는 발 미끄러짐 문제를 방지할 수 있습니다.

루트 모션은 고난도 콤보로 펀치를 여러 차례 날리는 복서 캐릭터처럼 발을 전후좌우로 빠르게 움직이는 복잡한 움직임에 적합합니다.

점프 애니메이션이 포함된 클립은 컷씬 애니메이션에 유용하게 사용할 수 있으며, Unity의 타임라인을 사용하여 캐릭터의 위치를 제어할 수 있습니다. 실제 게임플레이 중 점프 애니메이션의 경우 캐릭터의 Y 위치를 클립에 베이크하면 캐릭터가 공중에 뜨는 문제를 방지할 수 있습니다.



루트 모션 점프 클립의 Y 위치를 베이크합니다.

스크립트를 사용하여 캐릭터의 점프를 제어할 수 있습니다. 최고의 점프 애니메이션을 구현하려면 Rigidbody.AddForce를 사용하세요.

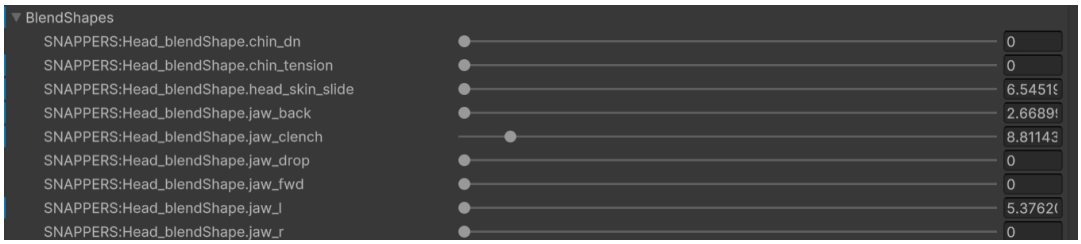
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

Unity Script (1 asset reference) | 0 references
public class JumpScript : MonoBehaviour
{
    private Rigidbody rb;
    public float jumpForce = 12f;
    private Animator animator;

    private

    // Start is called before the first frame update
    Unity Message | 0 references
    void Start()
    {
        rb = GetComponent<Rigidbody>();
        animator = GetComponent<Animator>();

    }

    // Update is called once per frame
    Unity Message | 0 references
    void Update()
    {
        if (Input.GetButtonDown("Jump"))
        {
            animator.SetTrigger("Jump");
            rb.AddForce(Vector3.up * jumpForce, ForceMode.Impulse);
        }
    }
}
```

코드에서 리지드바디를 사용하여 캐릭터의 점프 제어

블렌드 셰이프

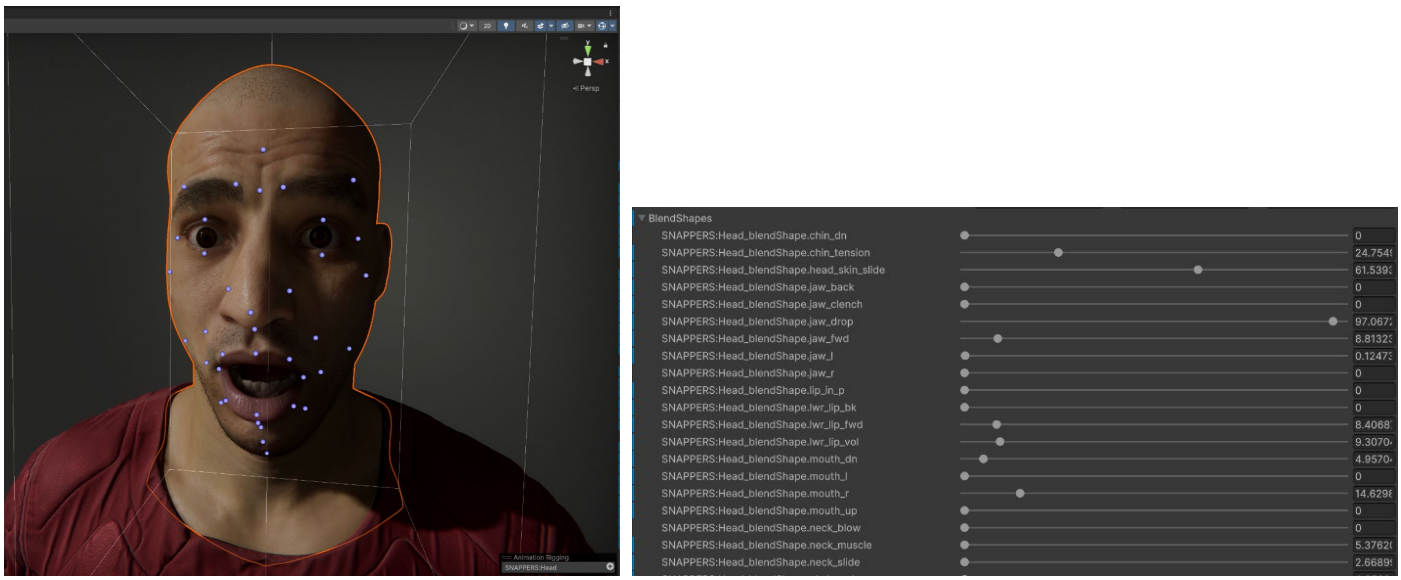
다른 3D 소프트웨어 툴에서는 모프 타겟이라고도 하는 블렌드 셰이프는 원본 메시 셰이프에 블렌딩할 수 있는 캐릭터 메시 셰이프의 버텍스 변형입니다.

블렌드 셰이프는 FBX 파일에 베이크되며 인스펙터에서 조정할 수 있습니다. 아래 이미지에 표시된 슬라이더로 블렌드의 가중치를 설정할 수 있으며 0이 최소 가중치, 1이 최대 가중치입니다. 여러 슬라이더를 조합하여 새로운 블렌드 모프를 얻을 수 있습니다.



인스펙터의 블렌드 셰이프 슬라이더

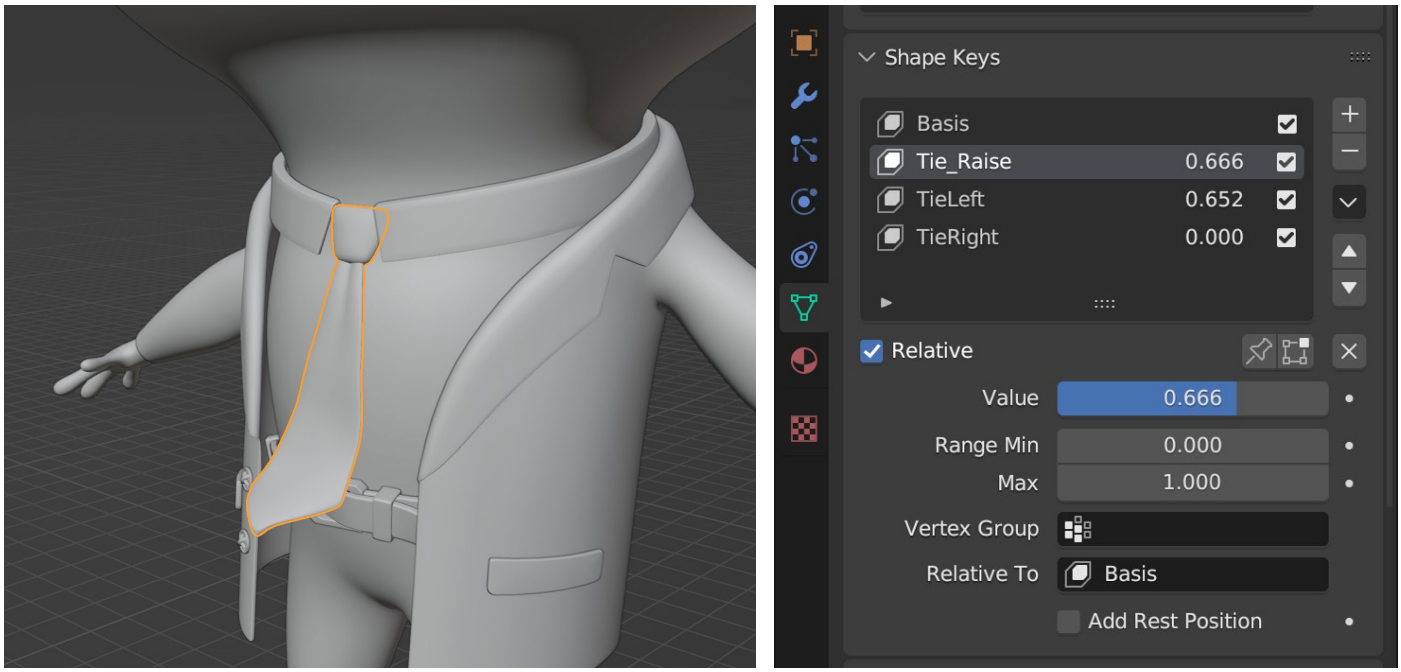
얼굴 릭 블렌드 셰이프는 더 헤러틱의 가웨인과 같은 복잡한 캐릭터에 흔히 사용됩니다. 얼굴의 점을 드래그하여 실시간으로 다양한 표정을 만들 수 있습니다. 여기에 주름 맵을 사용하는 디지털 휴먼 셰이더를 결합하면 얼굴 변형에 따라 사실적인 주름을 추가할 수 있습니다.



Unity 데모 더 헤러틱에서 디지털 휴먼 캐릭터의 얼굴에 적용된 블렌드 셰이프

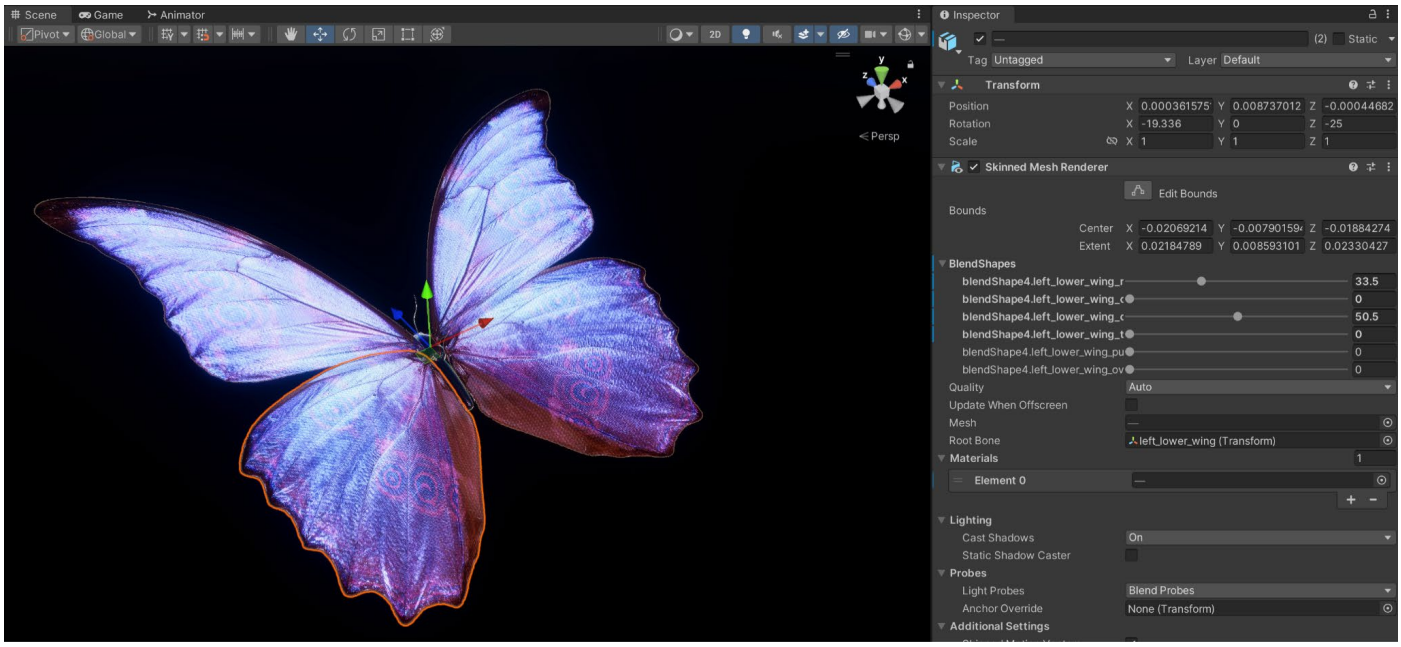
블렌드 셰이프를 통해 만든 다양한 표정

Unity에서는 천에 블렌드 셰이프를 추가하여 2차 동작을 구현할 수도 있습니다. 이러한 유형의 블렌드 셰이프는 Blender나 Maya 같은 3D 소프트웨어로 제작합니다.



Blender에서 만든 셰이프 키를 사용하여 넥타이가 움직이도록 애니메이션 적용

나비 애니메이션의 날개도 블렌드 셰이프를 사용하여 제작한 모션의 예입니다.



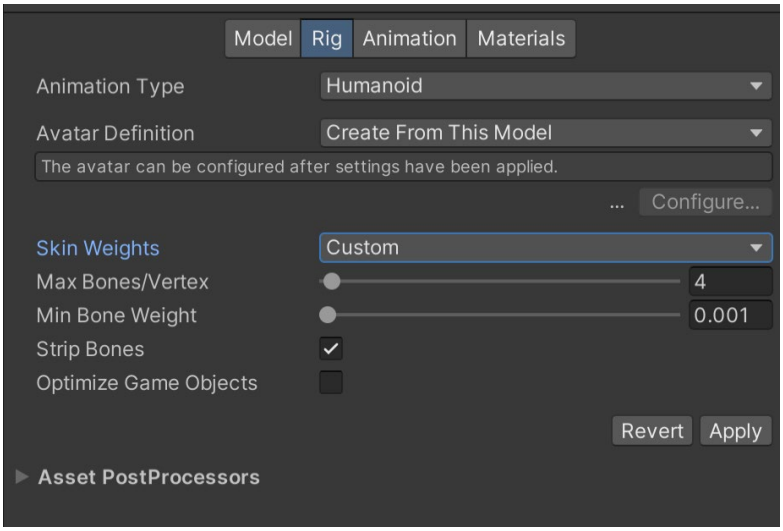
블렌드 셰이프를 통해 네 개의 날개 부위를 애니메이션화할 수 있음

휴머노이드 애니메이션 유형

휴머노이드 애니메이션은 대부분의 인간 캐릭터에 사용하는 애니메이션에 사용됩니다. 휴머노이드 애니메이션의 경우 Unity에서 직접 순운동학(FK)을 사용하여 키프레임을 설정할 수 없습니다. 다른 소프트웨어에서 애니메이션을 제작하여 Unity로 임포트할 수도 있습니다. 휴머노이드 애니메이션은 후반부 섹션에서 다룰 애니메이션 리깅을 통해 제어할 수 있습니다.

휴머노이드 애니메이션 유형은 회전 뼈대(twist bones)와 말단 뼈대(leaf bones) 같은 기능을 지원하므로 제네릭 유형보다 더 복잡합니다. 제네릭 애니메이션 유형처럼 명명된 뼈대에 의존하는 대신, 뼈대 위치에 대한 참조 지점을 사용하므로 휴머노이드 애니메이션은 대부분의 인간 캐릭터에 적용할 수 있습니다.

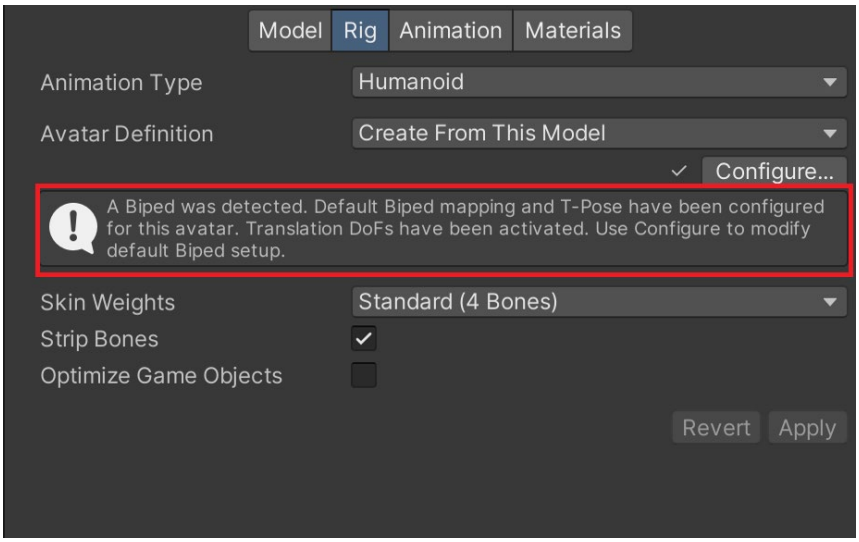
휴머노이드 애니메이션 유형을 설정하려면 캐릭터 모델이 T-포즈 또는 A-포즈여야 타겟을 가장 잘 매칭할 수 있습니다. FBX 모델 Rig 섹션의 드롭다운 목록에서 휴머노이드를 선택하고 이 모델에서 아바타를 생성합니다. **Apply**를 클릭하여 애니메이션 유형을 변경합니다.



캐릭터의 휴머노이드 릭 설정

Avatar Definition 필드에서 **Create from this Model** 옵션을 선택하면(위아래 이미지 참고) 휴머노이드 전환 프로세스가 선택된 캐릭터의 골격 구조를 분석합니다. 주요 위치의 뼈대를 식별하여 주요 참조 지점의 슬롯에 배치합니다. 예를 들어 상완 참조 지점은 캐릭터의 상완 뼈대를 사용합니다. 이제 각 뼈대의 명칭은 중요하지 않으며, 애니메이션은 참조 지점을 이용하여 골격을 애니메이션화할 수 있습니다.

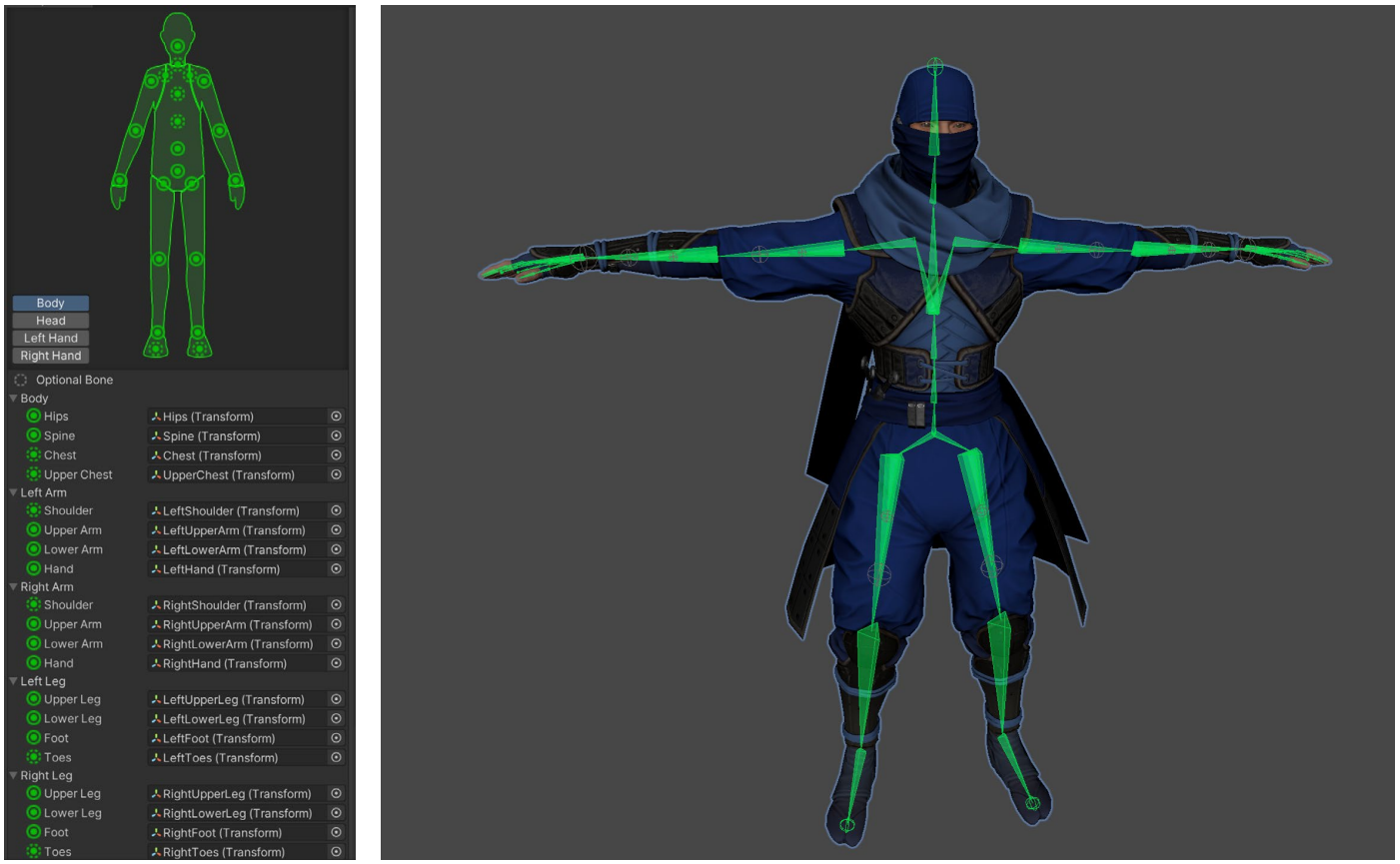
전환에 성공하면 Rig 섹션에 메시지가 표시됩니다.



휴머노이드 전환 성공

전환에 문제가 있다면 찾을 수 없는 뼈대가 Unity에서 강조 표시되며, 뼈대를 직접 매칭해야 할 수 있습니다. 캐릭터의 골격이 너무 복잡하면 휴머노이드로 전환하지 못할 수 있습니다.

Configure를 클릭하여 애니메이션 유형의 뼈대를 수동으로 설정합니다.



아바타의 주요 참조 지점에서 뼈대가 식별됩니다.

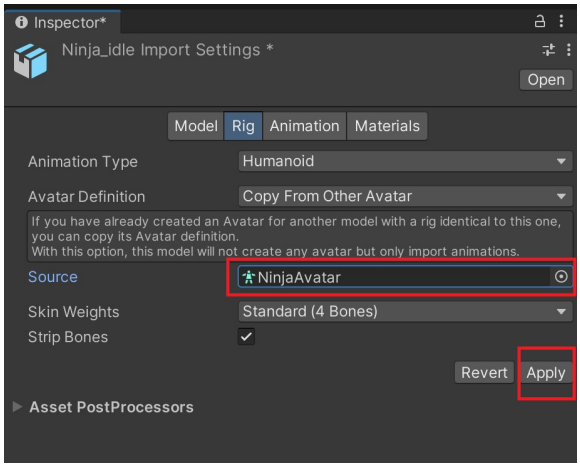
휴머노이드 애니메이션 유형의 캐릭터에 제네릭 애니메이션을 사용하면 캐릭터가 아래와 같은 설정 포즈를 취합니다.

휴머노이드 애니메이션 유형을 사용하는 캐릭터에서 제네릭 애니메이션이 재생되면 설정된 포즈를 취합니다.

캐릭터가 T-포즈 또는 A-포즈를 취해야 Unity에서 예상하는 위치에서 뼈대를 찾을 수 있으므로 휴머노이드 애니메이션 유형으로 전환하기 가장 좋습니다. 캐릭터가 애니메이션이 적용된 포즈를 취하고 있는 경우 뼈대가 평소와 다른 위치에 있어 전환이 어려울 수 있습니다.

T-포즈의 휴머노이드 아바타를 생성한 다음 이 아바타를 사용하여 임포트하는 모든 애니메이션 클립 파일을 전환하세요.

예를 들어 Mixamo에서 애니메이션 클립을 임포트하는 경우 전환 프로세스에서 Avatar Definition 필드를 **Copy From Other Avatar** 옵션으로 설정하고 T-포즈 캐릭터의 아바타를 선택하면 됩니다.



애니메이션을 휴머노이드 릭으로 전환하고 T 포즈 캐릭터의 아바타 사용하기

T-포즈 캐릭터 아바타의 골격 구조가 애니메이션 클립 파일과 동일한 것이 중요하므로, 애니메이션을 제작하거나 다운로드하기 위해 사용하는 소프트웨어에서 애니메이션 클립과 함께 T-포즈 모델도 임포트해야 합니다.

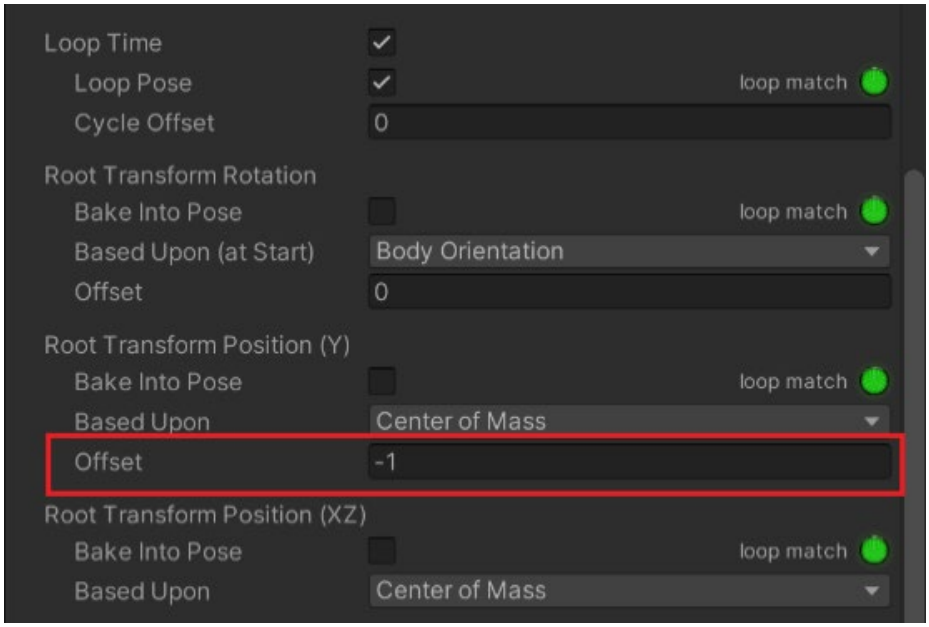
이러한 클립을 휴머노이드로 전환할 때 다른 T-포즈 캐릭터의 아바타를 사용하면 문제가 발생할 수 있으니 주의해야 합니다.

클립을 휴머노이드 애니메이션 유형으로 전환하면 이제 원본 캐릭터는 물론 휴머노이드 애니메이션 유형을 사용하는 대부분의 이족 보행 캐릭터에서도 잘 작동합니다.

골격 구조가 다른 3개의 캐릭터에 임포트하여 사용된 걷기 애니메이션

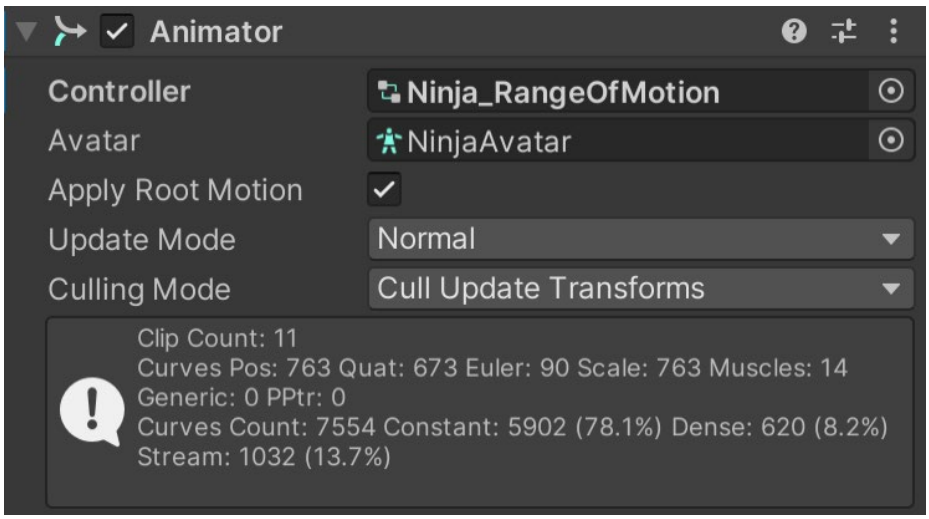
휴머노이드 클립은 다른 클립과 매칭을 위해 트랜스폼과 회전 오프셋을 조정할 수 있는 등 제네릭 클립보다 더 많은 설정이 있습니다.

아래 이미지에서 볼 수 있듯이 모든 설정의 **loop match** 옵션이 초록색으로 표시됩니다. 이는 첫 키프레임과 마지막 키프레임이 매칭되어 부드럽게 반복된다는 의미입니다. 트랜스폼의 경우 애니메이션에 루트 모션 키프레임이 설정되지 않아 캐릭터가 제자리에서 애니메이션을 재생한다는 의미일 수 있습니다. 이러한 유형의 클립은 코드를 사용하여 캐릭터를 움직일 때 유용합니다.



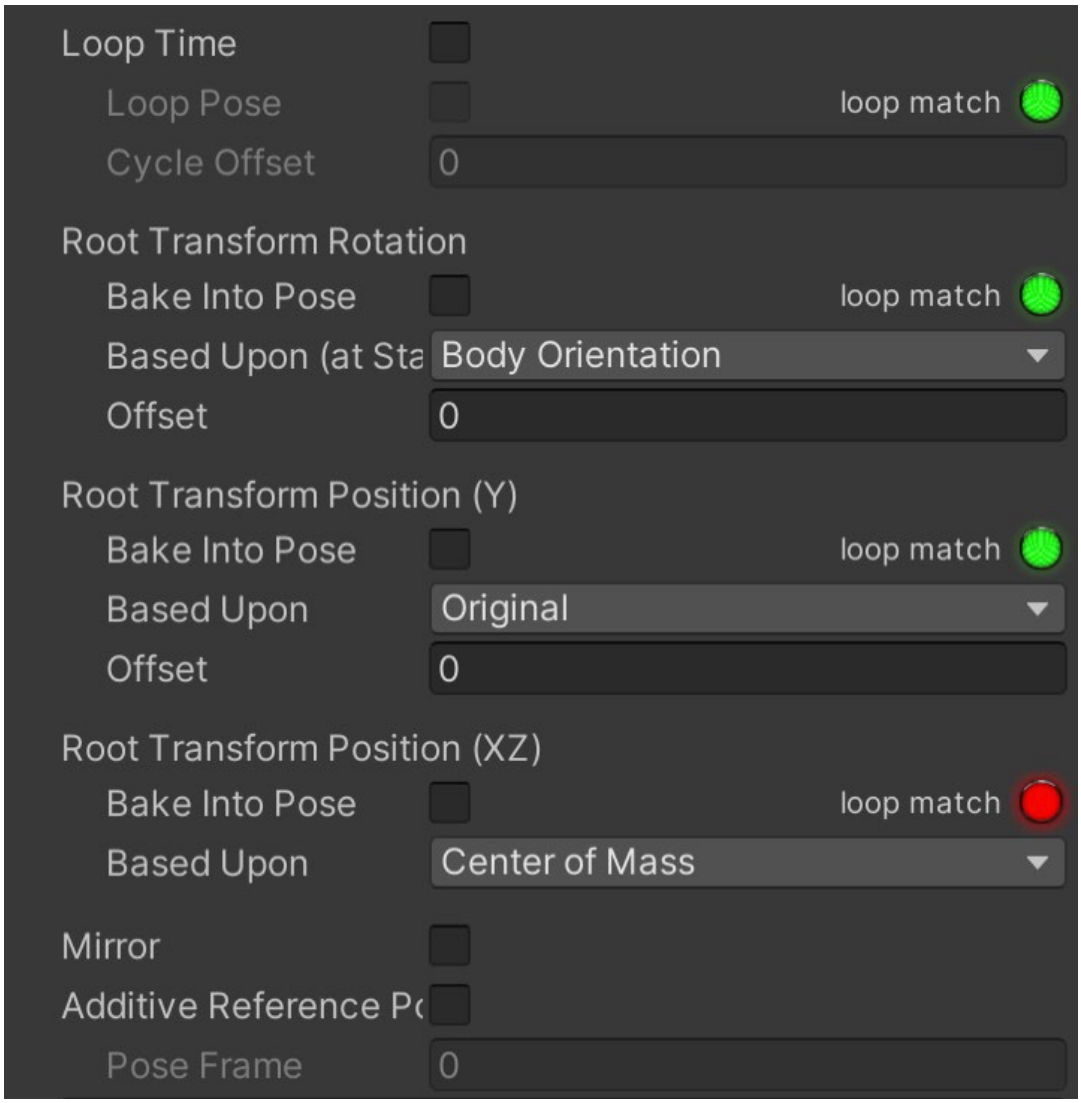
애니메이션 설정 및 오프셋 사용

빨간색 점은 키프레임이 일치하지 않아 애니메이션이 반복될 때 부자연스럽게 구현된다는 것을 나타냅니다. 모든 클립이 반복을 염두에 두고 설계되지는 않지만, 빨간색으로 표시되면 애니메이션에 모션 키프레임이 설정되었다는 의미일 수 있습니다. 이는 루트 모션이 사용된 것이며 휴머노이드 애니메이션에 루트 모션을 사용하는 경우 Animator 컴포넌트에서 **Apply Root Motion** 옵션을 활성화해야 합니다.



루트 모션을 적용하여 루트 모션 클립 사용

리지드바디 물리를 사용하는 캐릭터는 루트 모션 파일과 충돌이 일어날 수 있습니다. 이러한 충돌을 해결하려면 **Bake Into Pose** 옵션을 사용하여 클립에 모션을 베이크합니다.



루트 모션과 Bake Into Pose 옵션을 사용하는 루프

위아래 점프 모션을 처리하기 위해 **Root Transform Rotation**과 **Root Transform Position (Y)** 설정에서 이 옵션을 활성화합니다. 그리고 전방 및 후방 걷기 모션을 유지하기 위해 **Root Transform Position (XZ)** 설정의 Bake Into Pose 옵션을 비활성화합니다.

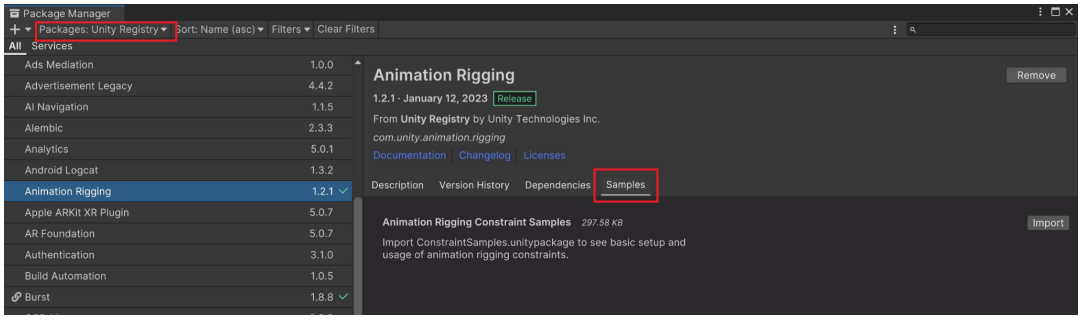
애니메이션 리깅

비디오 게임에서는 게임플레이 도중 캐릭터의 환경이나 위치가 변합니다. 캐릭터가 월드에서 사실적으로 움직이게 하는 애니메이션에 필요한 배리에이션을 모두 예상할 수 없는 경우도 많습니다. 따라서 런타임에 게임 변수나 특정 요구 사항에 따라 기존 애니메이션을 조정할 수 있는 것은 반드시 필요하며 시간을 절약하는 길입니다.

Animation Rigging 패키지를 사용하면 Unity 내 모든 유형의 캐릭터 또는 오브젝트를 위한 역운동학(IK) 릭을 제작할 수 있습니다. 제네릭 및 휴머노이드 애니메이션 유형에 모두 사용할 수 있으며 타임라인 에디터에서도 사용할 수 있습니다.

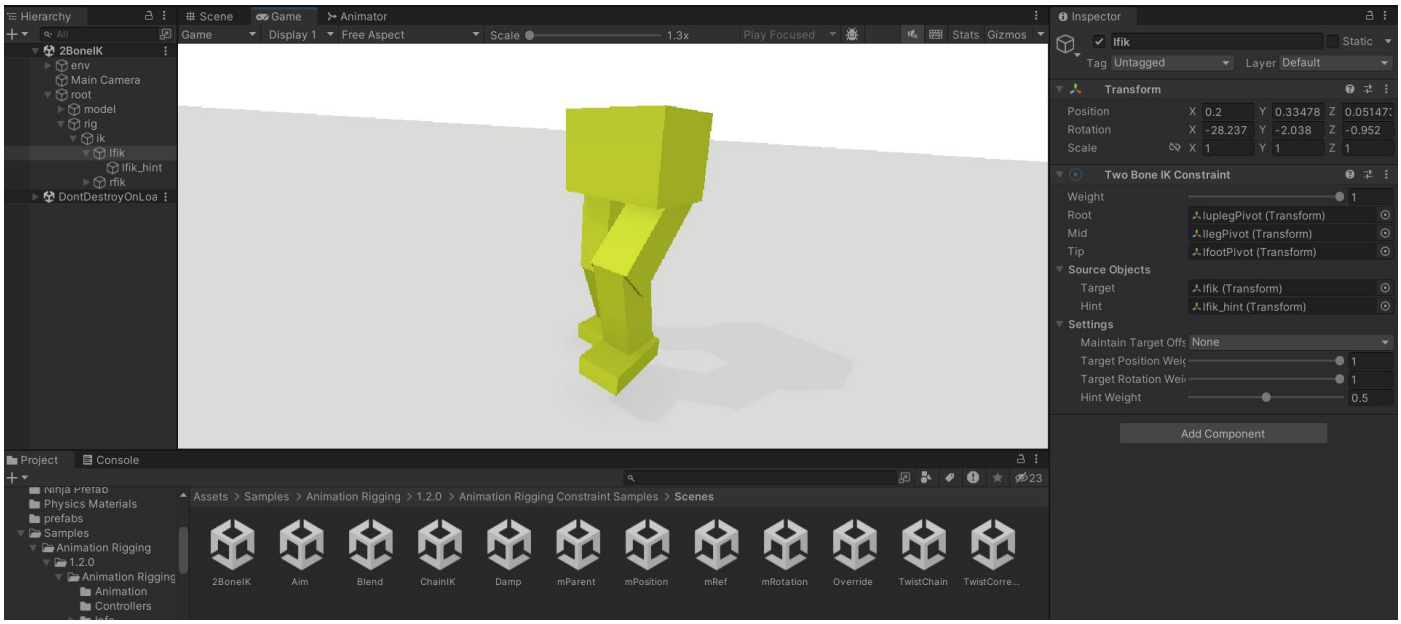
> **역운동학(IK)**은 골격 구조에 포함된 조인트의 위치 및 회전을 캐릭터의 손이나 발과 같은 말단 이펙터의 원하는 위치를 기반으로 결정하기 위해 애니메이션 및 로보틱스에서 사용되는 기법으로, 더 자연스럽고 사실적인 움직임을 구현할 수 있습니다. 게임에서 캐릭터의 손을 조작한다고 상상해 보세요. 역운동학(IK)을 사용하면 팔의 모든 조인트를 수동으로 움직여 원하는 위치로 손을 옮기는 대신 손을 배치할 위치를 지정하기만 하면 됩니다. IK 시스템은 손을 해당 위치로 옮기기 위해 자동으로 어깨, 팔꿈치, 손목 조인트를 계산 및 조정하여 더 자연스럽고 부드러운 움직임을 구현합니다.

Window > Package Manager로 이동하여 좌측 상단의 드롭다운에서 Unity Registry를 선택한 다음 Animation Rigging 패키지를 설치합니다. 패키지를 설치하면 각 리깅 제약의 예시 씬이 담긴 샘플 패키지를 다운로드할 것을 권장합니다. 해당 예시를 통해 리깅 제약의 프로젝트 구현 방법을 알아볼 수 있습니다.



패키지 관리자에서 Animation Rigging 패키지 설치

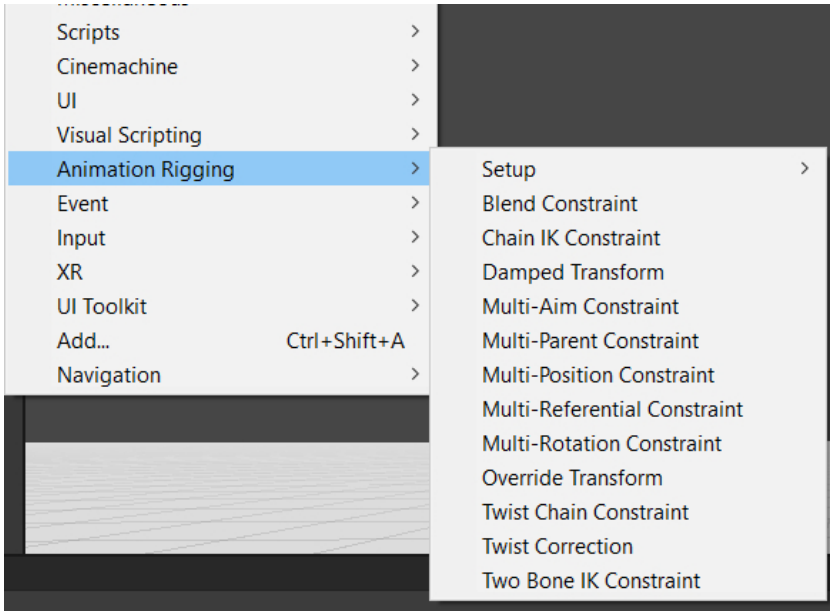
샘플 패키지는 Samples 폴더에서 찾을 수 있습니다. 플레이 모드를 실행하고 계층 구조에서 구조를 확인하여 어떻게 작동하는지 미리 확인해 보세요.



예시 씬은 각 제약이 어떻게 적용되는지를 보여 줍니다.

사전 정의된 스크립트 기반의 제약을 통해 Unity에서 IK 솔루션을 더 쉽게 설정할 수 있습니다. 또한 제약은 모듈식이니 캐릭터 골격의 모든 뼈대를 하나의 IK 릭으로 제어하는 대신, 동물이나 생물 같은 이족 보행이 아닌 캐릭터를 포함한 모든 캐릭터 유형에 맞게 여러 개별 제약을 혼합할 수 있습니다. IK 릭의 특정 부위를 나머지 부위와 독립적으로 끌 수도 있습니다.

제약은 **Component > Animation Rigging**에서 액세스할 수 있습니다.

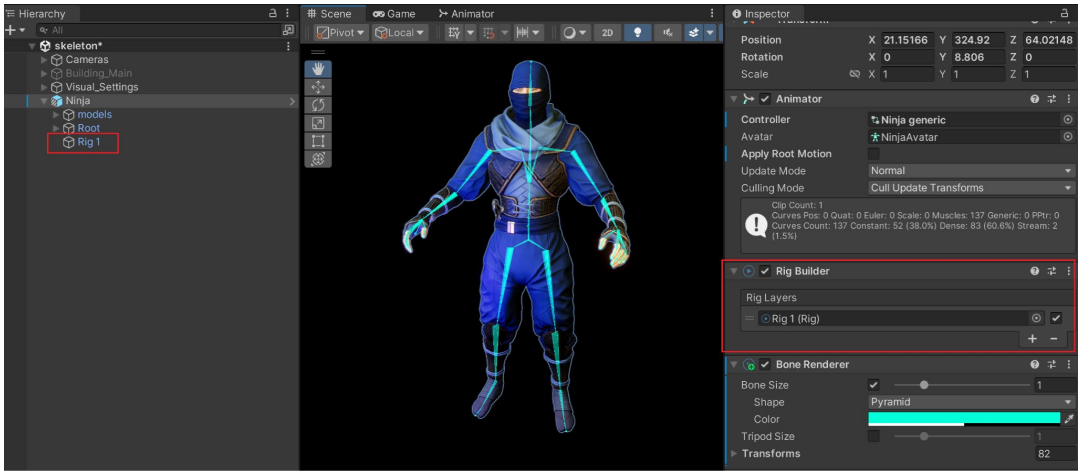


리깅 제약

IK 릭 설정

캐릭터를 선택하고 **Animation Rigging > Rig Setup**으로 이동합니다. Animator 컴포넌트와 Rig Builder 컴포넌트가 추가됩니다. 루트 캐릭터 게임 오브젝트에 Rig Builder 컴포넌트를 추가하면 자동으로 해당 루트의 자식 릭 게임 오브젝트가 생성되며, 기본 이름은 Rig 1이고 변경할 수 있습니다(아래 이미지 참고).

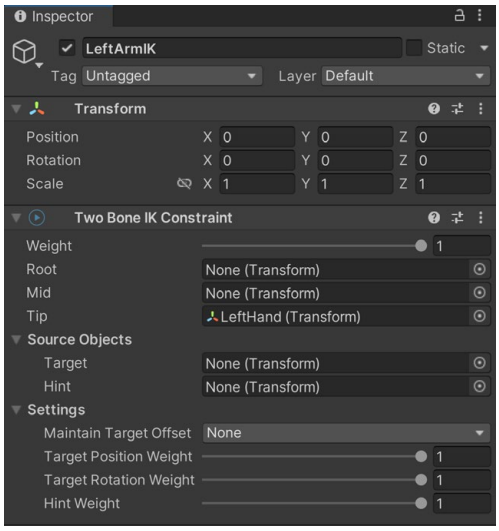
Animation Rigging > Bone Renderer Setup으로 이동하여 뼈대 레이아웃을 표시할 수도 있습니다. 이렇게 하면 골격 레이아웃이 시각적으로 표시되지만 리깅 패키지를 실행하는 데 반드시 필요한 작업은 아닙니다. 이 가이드에 나와 있는 예시의 나머지 부분에서는 캐릭터가 더 잘 보이도록 Bone Renderer 컴포넌트를 제거했습니다.



닌자 캐릭터의 IK 릭 설정

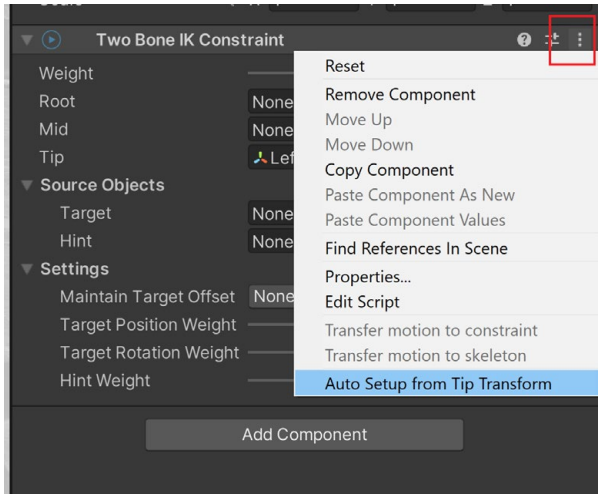
모듈식 릭 컴포넌트를 추가하려면 빈 게임 오브젝트 'Rig 1'을 추가합니다. 그런 다음 컴포넌트 메뉴에서 추가할 제약을 선택할 수 있습니다.

Two Bone IK Constraint는 팔, 다리, 손가락에 역운동학적 제약을 추가하기 위해 사용됩니다. 계층 구조에서 손을 드래그하여 **Tip** 슬롯에 놓습니다.



Tip 오브젝트를 Two Bone IK Constraint에 추가

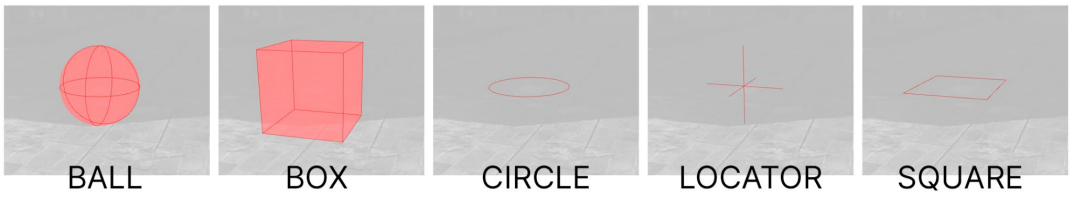
점 3개 옵션 아이콘을 클릭하고 **Auto Setup from Tip Transform**을 선택합니다. 이렇게 하면 연결된 뼈대를 자동으로 찾고 올바른 위치에 타겟 및 힌트 오브젝트를 추가합니다.



Auto Setup from Tip Transform을 사용하면 Two Bone IK Constraint가 자동 설정됨

타겟 및 힌트 오브젝트를 시각화하기 위한 이펙터도 추가됩니다. 타겟 오브젝트는 손 위치와 회전을 제어하며 힌트 오브젝트는 팔꿈치 회전을 제어합니다.

다양한 이펙터를 선택할 수 있으며 이펙터의 크기, 위치, 회전, 색상을 변경할 수 있습니다.



타겟과 힌트를 시각화하는 데 사용하는 이펙터의 유형

캐릭터가 상자를 운반할 수 있도록 위치 지정된 Two Bone IK Constraint

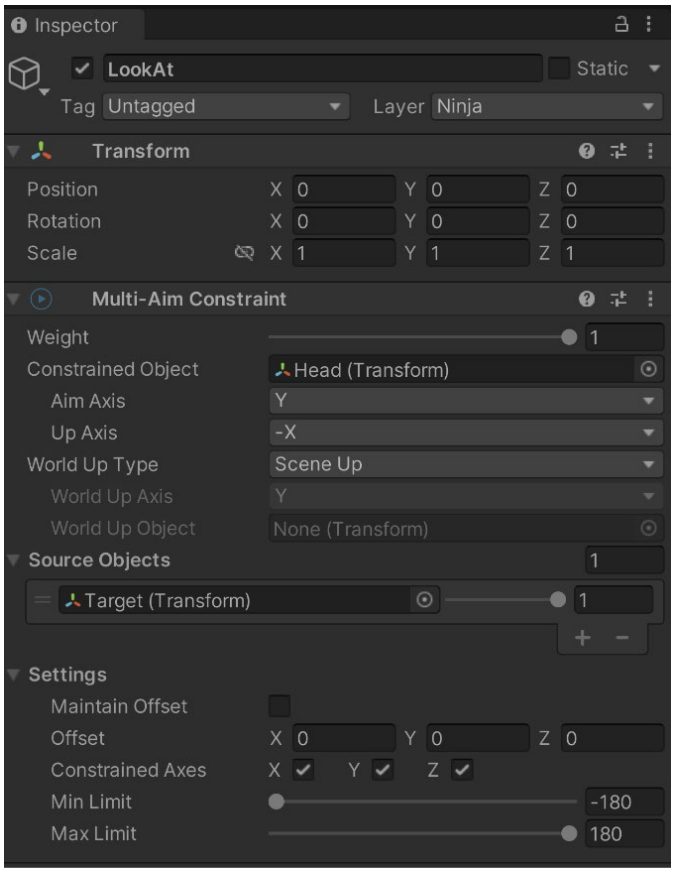
두 팔에 Two Bone IK Constraints를 적용했으니 이제 3D 공간에 위치를 지정할 수 있습니다. 이 애니메이션은 팔에 영향을 주지는 않지만 나머지 신체 부위는 계속 제어합니다. 이러한 방식은 캐릭터가 벽에 기대거나, 문을 열거나, 상자를 드는 등의 동작에 유용하게 사용할 수 있습니다.

모든 제약에는 가중치 슬라이더가 있어 타겟에 미치는 전반적인 영향을 조정할 수 있습니다. 강도를 1로 설정하면 타겟이 손 위치를 완전히 제어합니다. 강도를 0으로 설정하면 손에 아무런 영향을 주지 못합니다. 위치 가중치와 회전 가중치를 독립적으로 조정할 수도 있습니다.

플레이 모드나 녹화 모드에서 타겟을 이동하거나 회전시켜 캐릭터 팔의 위치를 변경해 볼 수 있습니다.

Multi-Aim Constraint를 사용하면 캐릭터의 머리가 움직이는 오브젝트 같은 타겟을 바라보도록 구현할 수 있습니다. 타겟이 해당 오브젝트의 부모가 되며 캐릭터는 오브젝트가 움직이는 것을 따라 보게 됩니다.

캐릭터가 움직이는 타겟을 바라보도록 Multi-Aim Constraint 사용



올바른 회전을 위해 **World Up Type** 필드에서 Scene Up 옵션을 사용하는 Multi-Aim Constraint 컴포넌트

머리가 올바르게 회전하는 것 같지 않으면 머리의 **Aim Axis**와 **Up Axis** 방향을 확인해 보세요. 이때는 로컬 공간에 있어야 합니다. 머리가 Y축을 따라 보고 있으며 Up Axis 필드는 -X로 설정되어 있습니다.

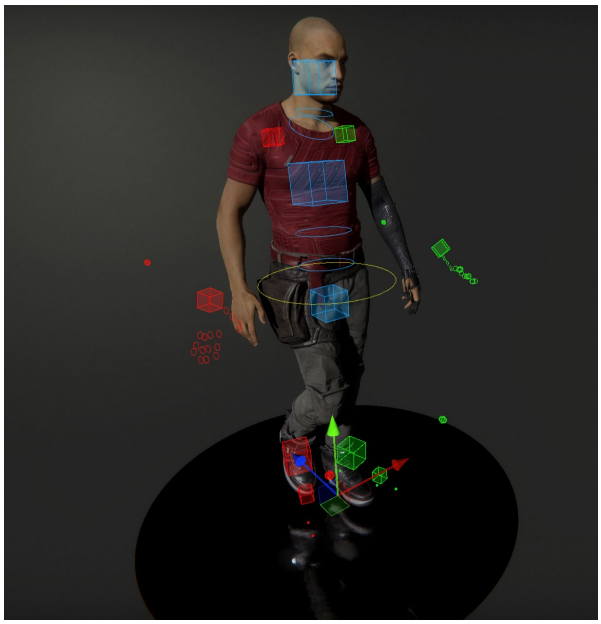
헤드는 Aim Axis에 양수 Y를, Up Axis에 음수 X를 사용하고 있습니다.

머리 회전을 -90도 및 90도로 제한하는 등 최소 및 최대 회전 값을 설정할 수도 있습니다.

다중 릭

Animation rigging > Rig setup에서 캐릭터에 원하는 만큼 릭을 생성할 수 있습니다. 두 번째 릭을 생성하면 Rig Builder 컴포넌트에 추가됩니다. Rig 1의 모든 제약은 Rig 2와 독립적으로 계산된다는 점에 유의하세요. 다중 릭은 2개 이상의 릭이 개별적으로 작동해야 하는 경우에 유용합니다.

풀바디 컨트롤 릭

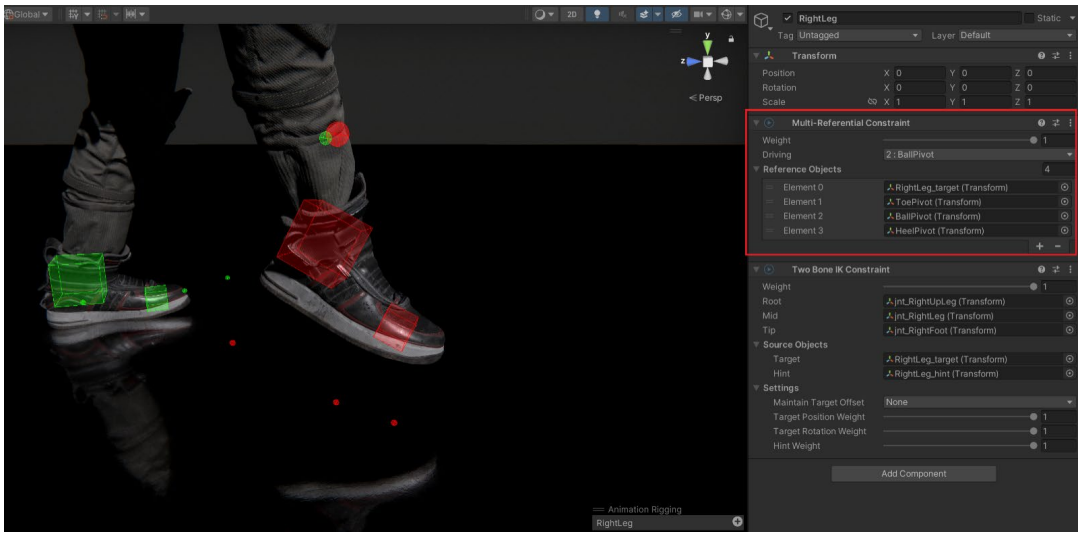


더 헤러틱 의 캐릭터 가웨인은 Multi-Parent, Multi-Position, Multi-Aim 등 다양한 제약을 활용하여 모든 신체 부위를 제어할 수 있는 풀바디 컨트롤 릭의 예입니다.

풀바디 컨트롤 릭 예시

이렇게 설정하면 어떤 상황에서도 캐릭터를 제어할 수 있다는 이점이 있습니다. 일부 제약은 캐릭터를 완전히 제어할 수 있는 IK이며, 다른 일부 제약은 기본 애니메이션에 영향을 주지 않으며 뼈대의 위치와 회전을 조정하는 오버라이드입니다.

특정 피벗에서 뼈대 위치를 제어할 수 있는 **Multi-Referential Constraint**를 사용하면 발 또는 손의 정밀한 위치 지정이 가능합니다. 발 위치 지정의 경우 발가락과 발볼을 사용하여 발의 정밀한 위치를 지정합니다.



Multi-Referential Constraint 컴포넌트를 사용하여 뼈대 피벗 조작

각 지점에 타겟 지점을 설정해야 하며 여기서는 발 뒤꿈치, 발볼, 발가락이 해당됩니다. 그런 다음 각 지점의 피벗에 의해 제어될 타겟인 오른쪽 다리 타겟과 함께 제약에 추가됩니다.

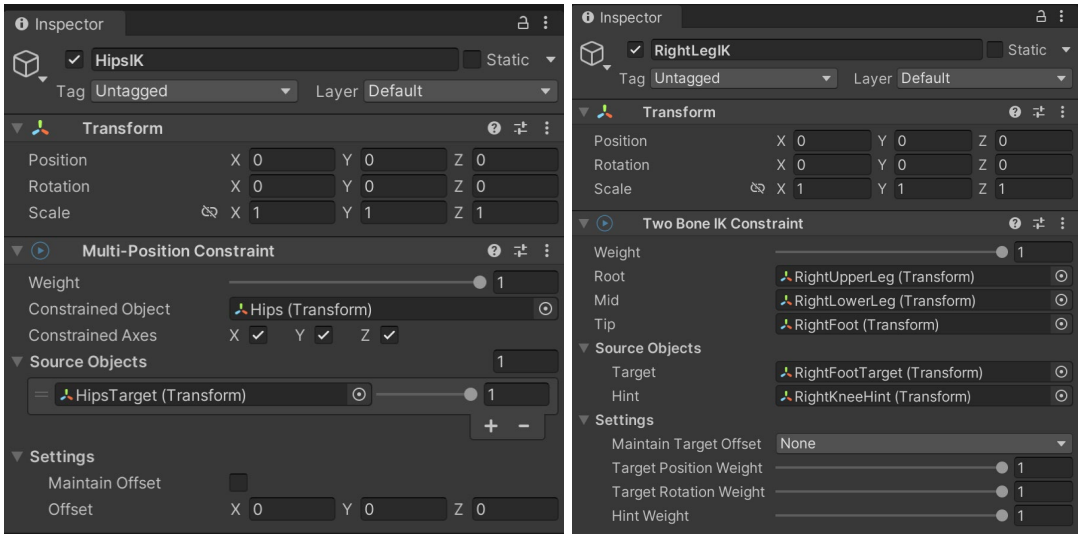
플레이 모드 또는 녹화 모드 중 드롭다운에서 이 피벗을 설정할 수 있습니다. 타겟 지점을 선택하고 회전시키면 해당 피벗 포인트에서 발이 회전하는 것을 볼 수 있습니다. 자세히 알아보려면 이 유나이트 세션을 참고하세요.

발 IK

발 IK는 사실적인 애니메이션을 제작하기 위한 핵심 개념이므로 대부분의 애니메이터에게 익숙한 기능입니다. 발 IK를 사용하면 발이 미끄러지거나 지면 아래로 가라앉는 문제를 줄일 수 있습니다. 캐릭터가 의자에 앉아 있을 때 발을 올바른 위치에 유지시키는 것이 그 예입니다.

아래 예시는 고관절을 제어할 수 있는 새로운 릭입니다. IK는 계층 구조에 표시되는 순서대로 계산되므로 고관절 제약을 두 개의 발 제약보다 상위에 놓는 것이 중요합니다. 그래야 고관절 움직임을 먼저 계산한 다음 발 위치를 계산합니다. 반대로 배치하면 발을 먼저 계산한 다음 고관절을 움직일 것입니다.

고관절 및 다리 IK



고관절 및 다리 IK 설정

두 개의 발은 캐릭터가 앉을 때 무릎을 올바르게 구부리도록 Two Bone IK Constraint를 사용합니다. 발은 항상 바닥에 붙어 있게 됩니다.

플레이 모드에서는 고관절 타겟의 위치가 지정되어 캐릭터가 의자에 앉습니다. 왼쪽 및 오른쪽 다리 힌트는 필요에 따라 무릎 위치를 조정하는 데 사용됩니다.

발을 바닥에 고정한 상태로 앉는 간단한 IK 애니메이션

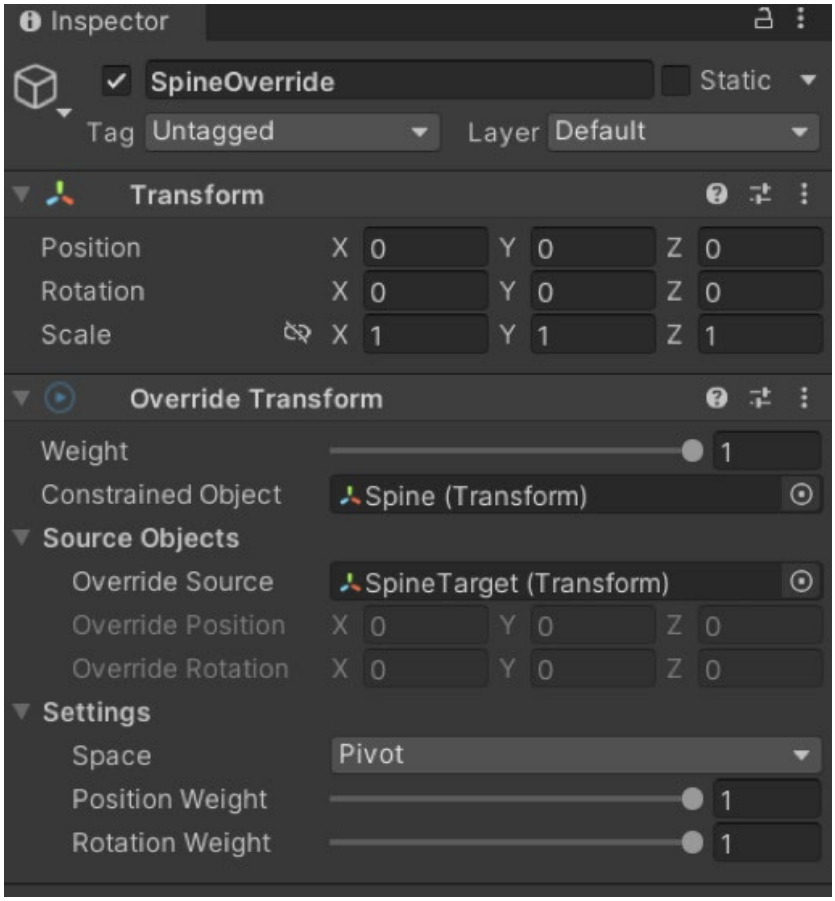
IK를 애니메이션과 병합

일반적으로 IK 컴포넌트는 신체 부위를 완전히 제어하므로 해당 부위에서는 애니메이션이 재생되지 않습니다.

하지만 **Override Transform** 컴포넌트를 사용하면 애니메이션과 IK 컨트롤을 병합할 수 있으며, 다른 신체 부위를 멈추지 않으면서 기존 애니메이션을 약간 조정하고 싶을 때 유용합니다.

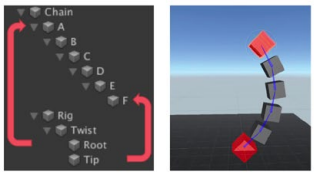
이를 설정하려면 각 뼈대 위치에 타겟을 설정해야 합니다. 가장 좋은 방법은 캐릭터 골격에서 척추 같은 뼈대를 오른쪽 클릭한 다음 빈 게임 오브젝트를 생성하는 것입니다.

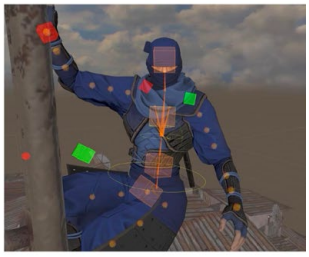
시각화를 위해 타겟에 이펙터를 할당하고 릭 섹션에 드래그합니다. 각 타겟에 Override Transform 컴포넌트가 추가되며 수정할 뼈대는 **Constrained Object** 필드에, 타겟 자체는 **Override Source** 필드에 추가됩니다.



애니메이션이 뼈대를 계속 제어하는 상태에서 오버라이드가 뼈대를 조정하게 됩니다.

이제 타겟에 애니메이션을 추가하여 애니메이션이 계속 재생되는 동안 척추 회전, 어깨 조정, 목 위치 지정, 다리 회전 등을 조정할 수 있습니다. 이 프로세스의 바람직한 예시를 보려면 이 동영상을 확인하세요.





프리폼 애니메이션 리깅: 애니메이션 파이프라인의 진화

이 GDC 연설에서 데이브 헌트는 유니티가 Animation Rigging 패키지에 프리폼 애니메이션을 구현하여 애니메이터들이 리깅 전문가의 도움 없이도 컨트롤 릭의 구조를 변경하며 모션 콘텐츠를 비파괴적으로 보존할 수 있도록 지원한 방법을 설명합니다.

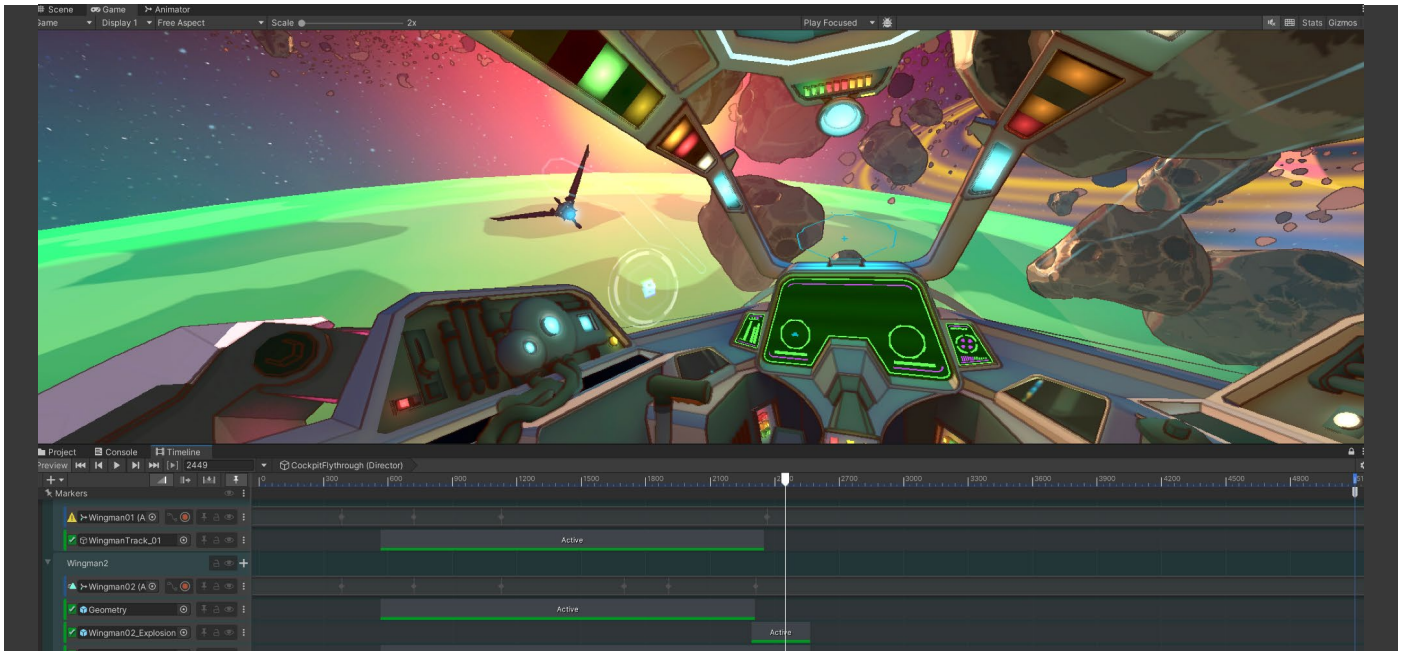
세션 시청하기

애니메이션 리깅의 장점

- IK 시스템은 모든 팀원이 쉽게 사용할 수 있습니다.
- Unity에서 직접 릭 작업을 반복하며 원하는 애니메이션을 얻을 수 있습니다.
- 한 캐릭터에 여러 제약이 있더라도 효율적인 재생 성능을 발휘합니다.
- 미리 보기 및 애니메이션을 위한 실시간 IK 업데이트를 통해 효율적으로 반복 작업하고 키프레임을 설정할 수 있습니다.
- 모듈식 릭 시스템으로 이족 보행 및 이족 보행이 아닌 캐릭터의 커스텀 IK 릭 설정을 생성할 수 있습니다.
- 모듈식 시스템을 사용하면 다양한 유형의 캐릭터를 위한 릭을 효율적으로 제작할 수 있습니다.
- 각 모듈식 릭 섹션의 가중치를 통해 시스템을 커스터마이즈할 수 있습니다.
- 동일한 릭 설정을 다른 캐릭터에 복사해 사용할 수 있습니다.
- 코드는 오픈 소스이며 개발자들이 수정할 수 있습니다.
- 타임라인을 사용하면 여러 제약 오버라이드를 통해 정교한 결과를 얻을 수 있습니다.
- 양방향 모션 전달을 통해 캐릭터 골격에 모션을 다시 베이크하거나 골격에서 IK 컨트롤 릭으로 모션을 베이크할 수 있습니다.

애니메이션 컷씬을 위한 타임라인

타임라인은 Unity에서 직접 선형 애니메이션 시퀀스를 제작할 수 있으며 동영상 에디터처럼 작동하는 애니메이션 에디터입니다. 한곳에서 씬 내 모든 오브젝트의 애니메이션을 제어할 수 있으며, 다른 오브젝트와 캐릭터에 따라 애니메이션이 재생되는 시점을 제어할 수도 있습니다.

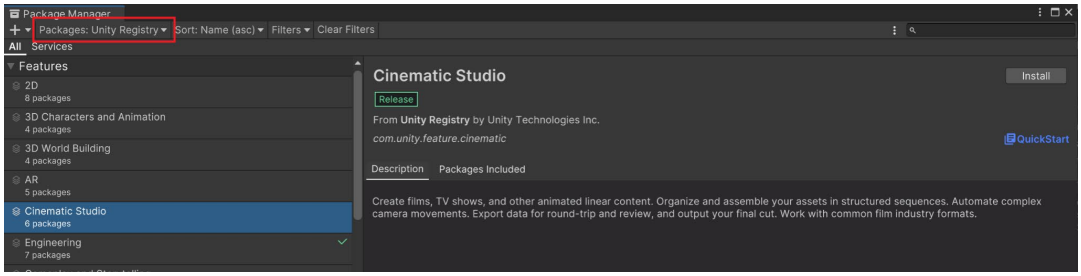


URP 3D 샘플 조종석 환경의 타임라인 예시

제네릭 및 휴머노이드 애니메이션 유형에서 모두 사용할 수 있고 Animation Rigging 패키지를 지원합니다.

패키지 관리자의 Cinematic Studio 컬렉션에 애니메이션 시퀀스를 제작하는 데 도움이 되는 다양한 패키지가 있습니다. **Window > Package Manager**에서 설치할 수 있습니다.

다음과 같은 패키지를 제공합니다.



패키지 관리자의 Cinematic Studio 패키지

— Timeline: 컷씬 같이 재생할 수 있는 시퀀스를 생성합니다.

— Sequences: 영화나 시네마틱 같은 선형 콘텐츠를 제작할 수 있습니다.

— FBX Exporter: Unity에서 제작한 지오메트리, 광원, 카메라, 애니메이션을 3D 모델링 소프트웨어에 전송할 수 있습니다.

— Alembic: Unity에서 애니메이션을 재생할 수 있도록 애니메이션 데이터를 파일에 베이크합니다.

— Cinemachine: 타겟 추적, 구성, 블렌딩, 샷 분할 등의 복잡한 수학과 로직을 해결하는 고급 카메라 시스템입니다.

— Recorder: 트레일러, 튜토리얼 및 기타 유형의 콘텐츠에 적합한 최적의 프레임 속도로 게임플레이 영상을 캡처하여 동영상 파일 또는 이미지 시퀀스로 제작합니다.

애니메이션 컷씬

게임의 컷씬은 Autodesk 3ds Max, Maya, Blender 같은 소프트웨어로 제작한 다음 동영상 파일로 렌더링하여 컷씬에서 재생하는 방식이 전통적이었습니다. 이러한 동영상 파일은 용량이 큰 탓에 동영상 인코딩을 통해 압축해야 했으므로 전반적인 품질이 저하될 수 있었습니다.

그러나 이제 Unity에 Cinemachine과 타임라인 시스템이 추가되어 동영상을 압축할 필요 없이 높은 품질의 컷씬을 게임 엔진에서 곧바로 재생할 수 있습니다.

에너미즈는 Unity에서 타임라인을 사용하여 제작된 단편 애니메이션입니다.

씬에 타임라인 시퀀스 추가

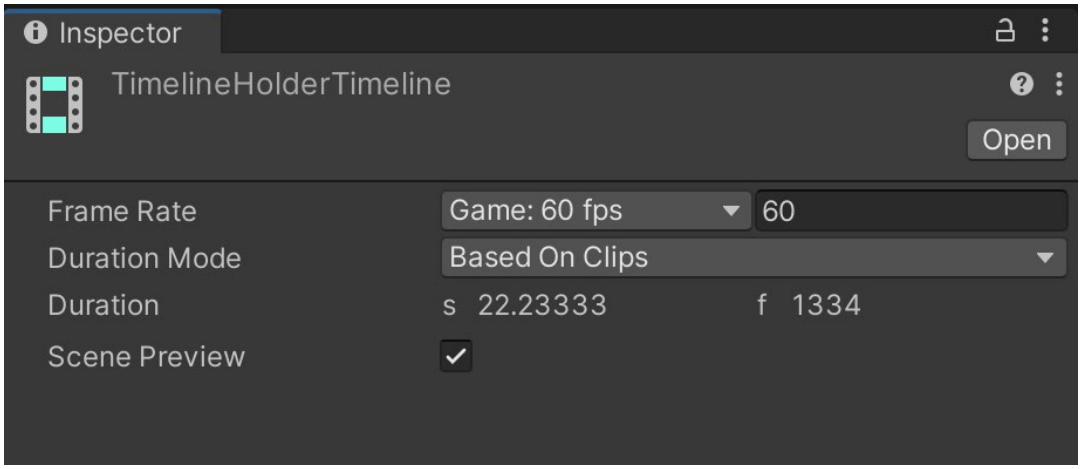
타임라인 시퀀스는 현재 씬의 일부로 추가하고 한 위치에 저장해야 합니다. 먼저 빈 게임 오브젝트를 생성하고 이름을 TimelineHolder로 설정합니다. **Window > Sequences > Timeline**에서 Timeline 창을 엽니다.

계층 구조에서 TimelineHolder 오브젝트를 선택하고 타임라인 에셋을 생성합니다. 여기에 애니메이션 클립을 포함한 모든 타임라인 정보가 저장됩니다.

또한 타임라인을 TimelineHolder 게임 오브젝트에 바인드하는 Playable Director 컴포넌트도 생성됩니다. 이제 계층 구조에서 TimelineHolder를 선택하면 트랙 에디터를 사용할 수 있습니다.

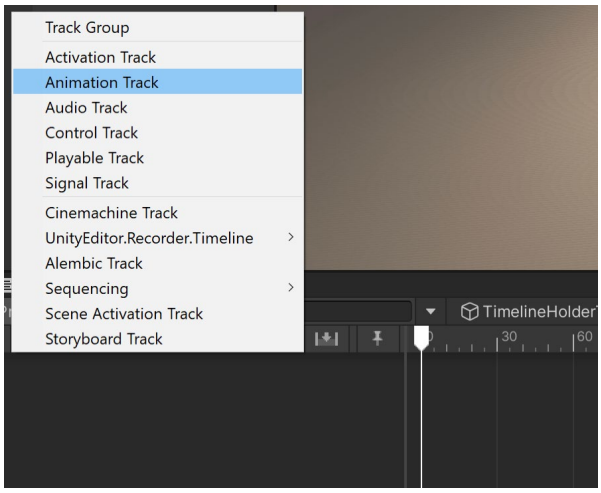
프로젝트 창에서 타임라인 에셋을 클릭하면 타임라인 시퀀스의 프로퍼티를 정의할 수 있습니다. 예를 들어 프레임 속도는 기본적으로 60fps지만 다른 프레임 속도로 변경할 수 있습니다.

타임라인 시퀀스의 지속 시간을 구체적으로 설정하거나 추가한 클립의 수에 따라 늘릴 수도 있습니다.



타임라인 프로퍼티

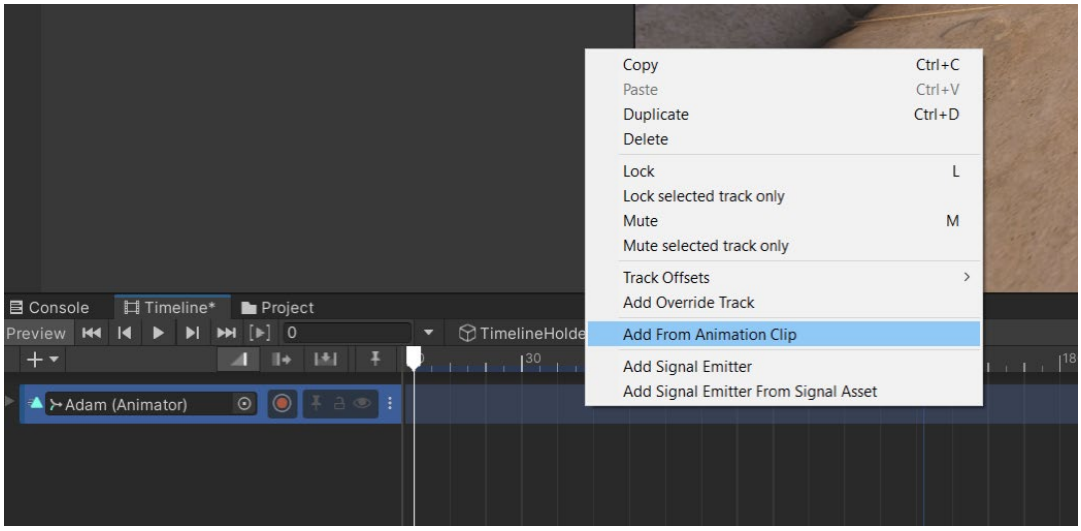
애니메이션 클립 추가



애니메이션 트랙 추가

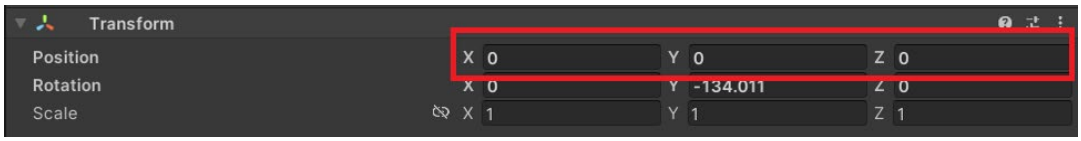
Timeline 창의 '더하기' 아이콘을 클릭하여 트랙을 추가하고 **Animation Track** 유형을 선택합니다.

애니메이션 클립을 트랙에 드래그하고 클릭 및 드래그하여 위치를 지정합니다. 트랙을 오른쪽 클릭하고 Add from Animation Clip을 선택한 다음 애니메이션을 선택할 수도 있습니다.



오른쪽 클릭하고 **Add from Animation Clip** 옵션을 선택하여 애니메이션 클립을 지정합니다.

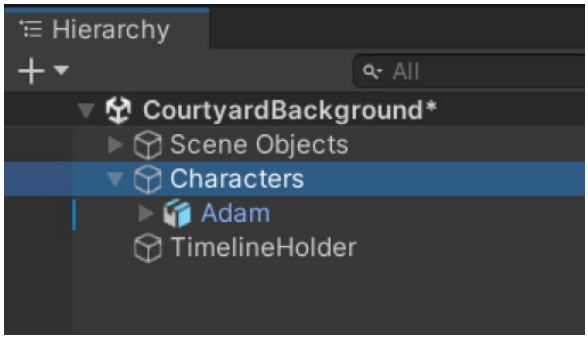
이때 캐릭터의 루트 원점이 월드 0인 경우 캐릭터가 사라질 수 있습니다. 캐릭터가 월드 X, Y, Z가 0인 위치로 이동하여 애니메이션을 재생하는 것입니다. 이 경우 캐릭터의 위치가 원하는 위치와 멀리 떨어져 있을 수 있습니다.



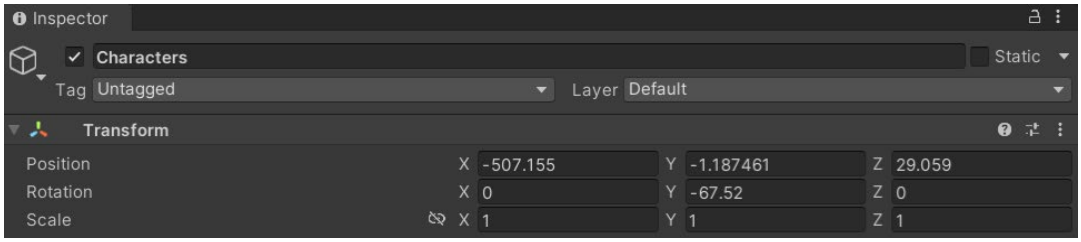
루트 원점이 0이면 캐릭터가 월드 위치 0으로 이동

이를 해결하려면 캐릭터의 현재 위치에 빈 게임 오브젝트를 생성하고 캐릭터를 안에 배치하여 부모로 만듭니다.

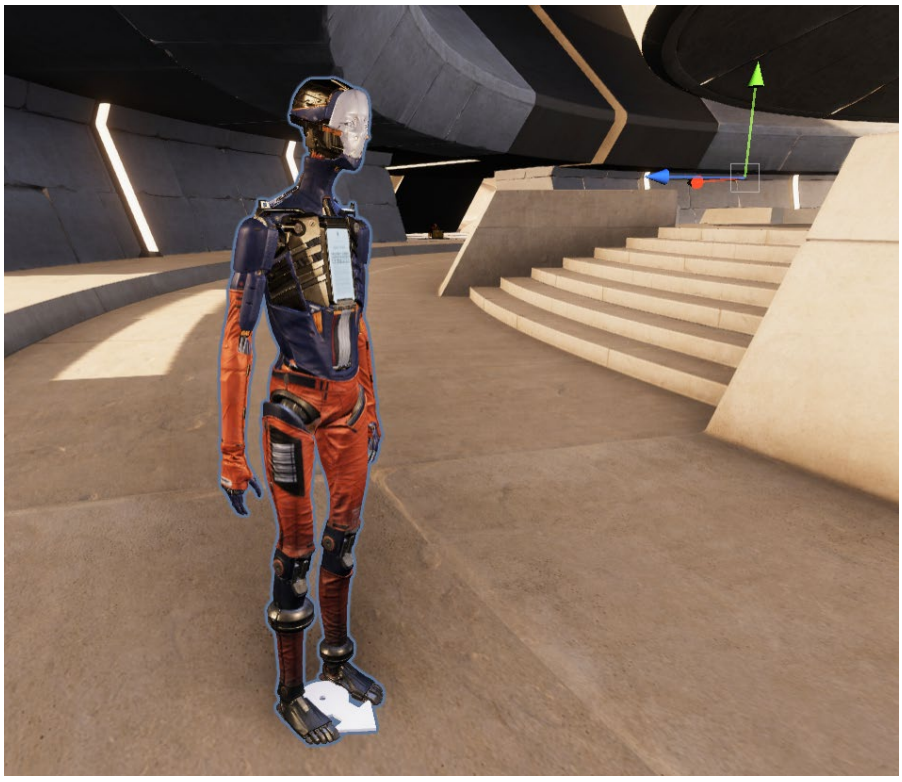
자식 오브젝트는 부모 오브젝트의 위치를 기준으로 합니다. 따라서 자식 오브젝트는 여전히 트랜스폼 X, Y, Z 축의 위치가 0이지만 월드 공간의 루트 원점인 0이 아니라 빈 게임 오브젝트의 현재 위치에 배치되는 것입니다.



이제 Adam 캐릭터는 Characters라는 빈 게임 오브젝트의 자식입니다.

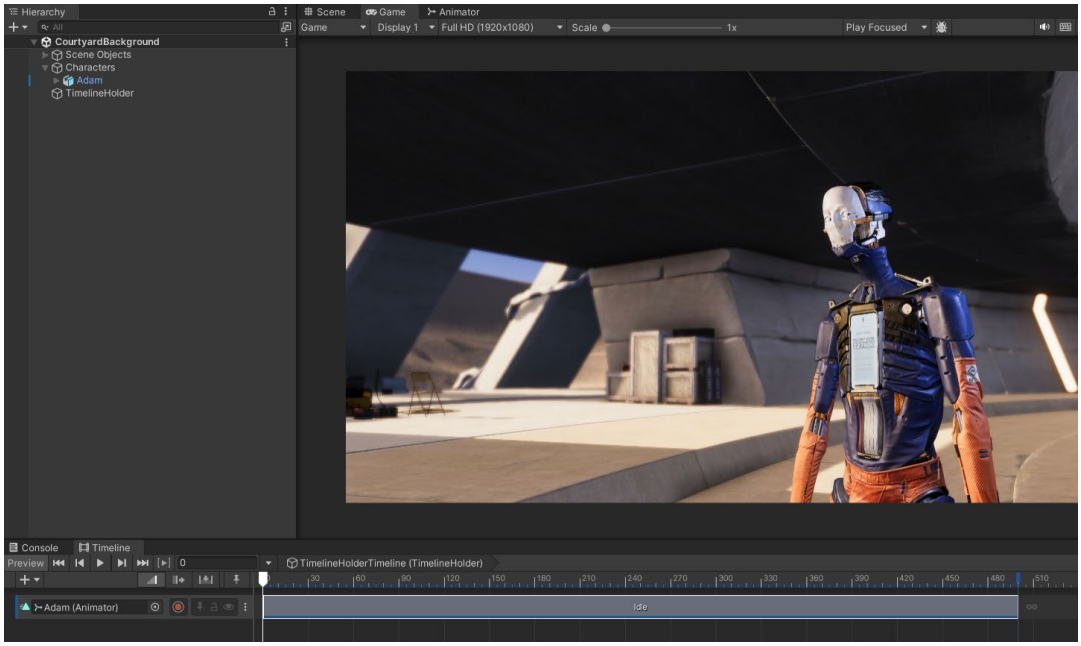


빈 Characters 게임 오브젝트의 위치

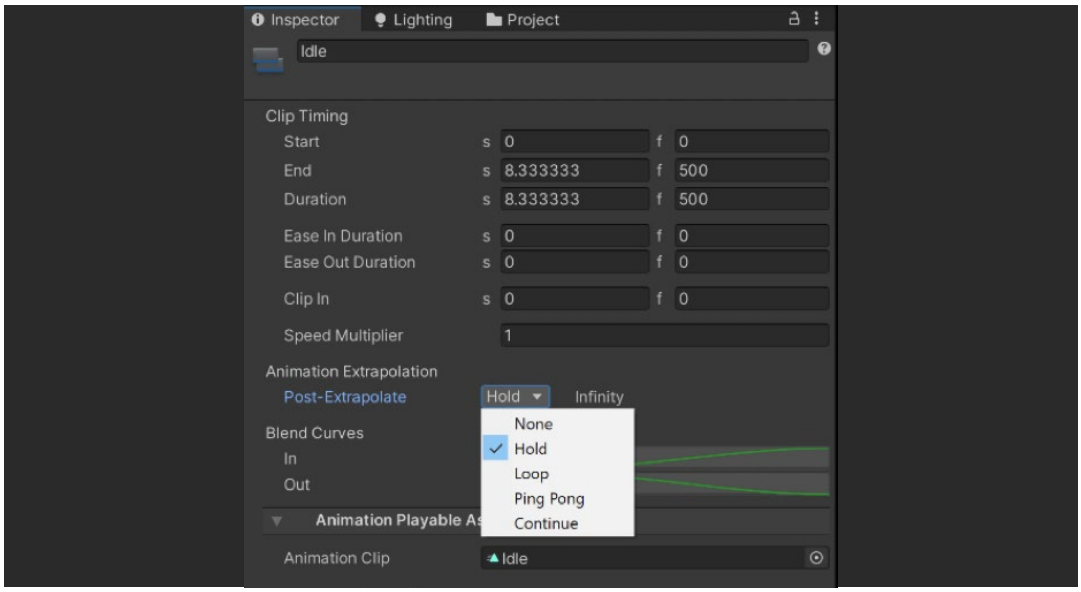


이제 Adam 캐릭터는 올바른 위치에서 애니메이션을 재생합니다.

Timeline 창의 트랙에서 애니메이션 클립은 불투명 블록으로 표시되며 트랙 안에서 시간을 앞뒤로 드래그할 수 있습니다.



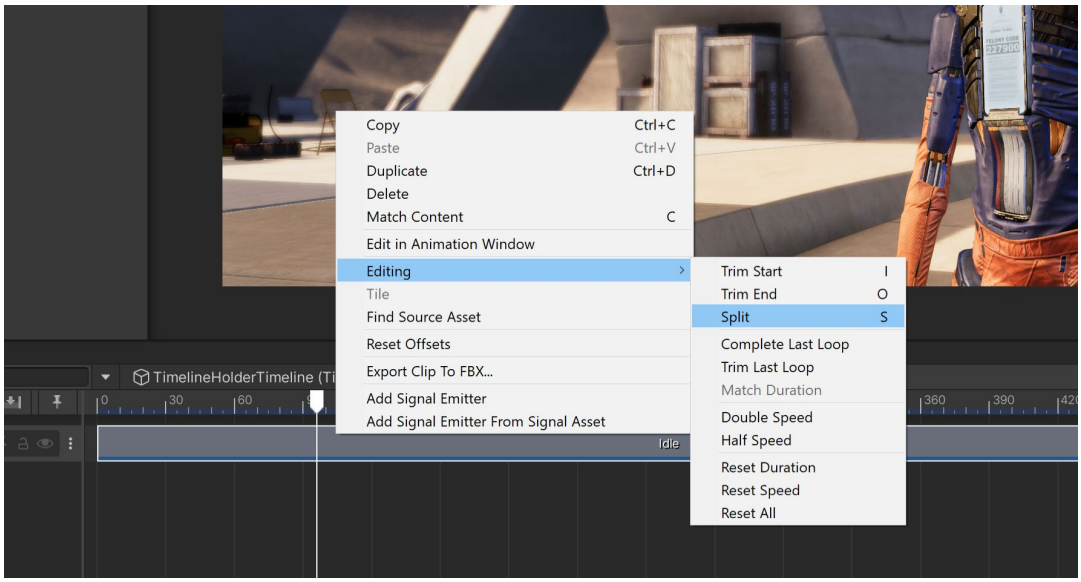
타임라인의 Adam 캐릭터 애니메이션 클립과 인스펙터의 클립 옵션



애니메이션 클립의 사후 외삽

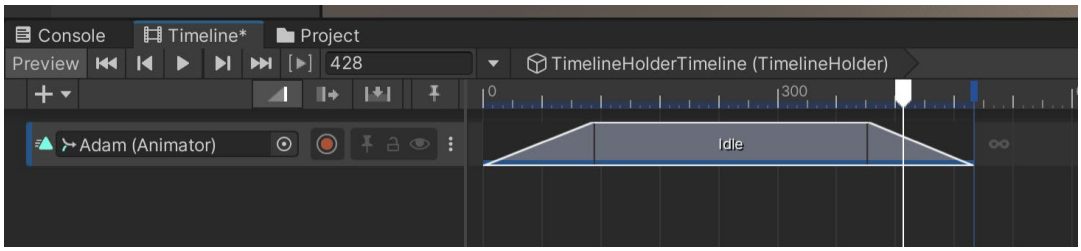
모든 클립은 전후에 공백이 있는 경우 사전 및 사후 외삽(extrapolation) 방식이 존재합니다. 이는 애니메이션 재생 전후에 무엇이 발생하는지 결정합니다. 기본적으로는 **Hold**로 설정되어 애니메이션의 첫 프레임이나 마지막 프레임의 포즈를 유지합니다.

클립을 오른쪽 클릭하고 Editing을 선택하여 동영상 에디터처럼 애니메이션 클립을 자르거나 그 크기를 조절할 수 있습니다. 단축키도 표시되므로 더 효율적인 편집이 가능합니다.

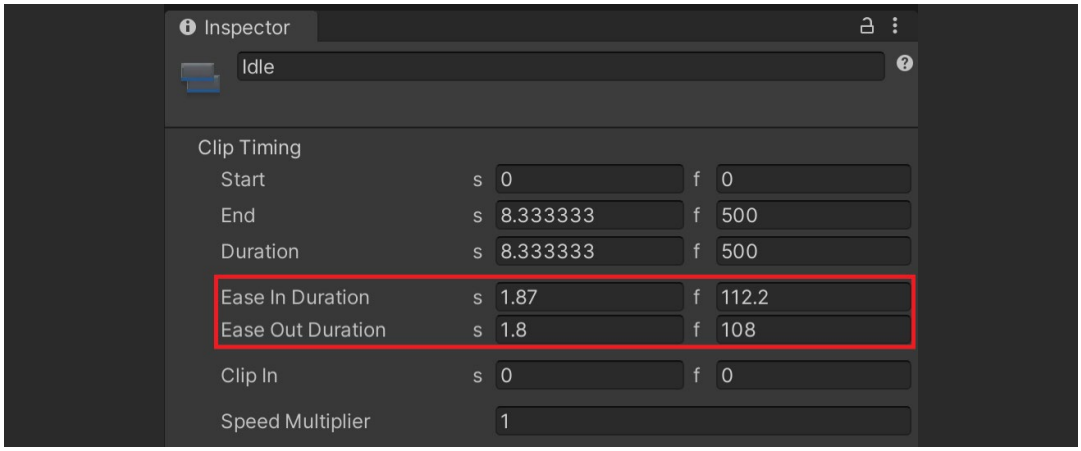


애니메이션 클립의 편집 옵션

클립 전후에 다른 클립이 없는 경우 페이드인 및 페이드아웃을 설정할 수도 있습니다. 이렇게 인스펙터에서 블렌드인 및 블렌드아웃 값을 조정하여 포즈를 서서히 변경하면 부자연스러운 구현을 방지할 수 있습니다.

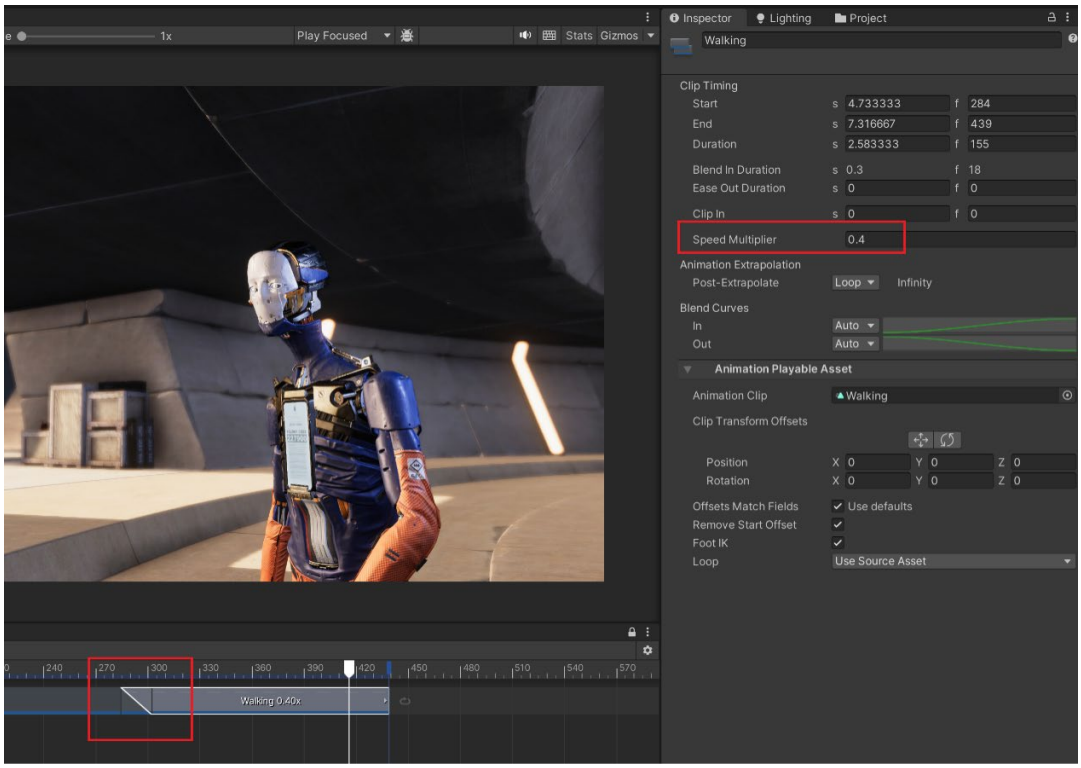


애니메이션 클립의 페이드인/페이드아웃 지점

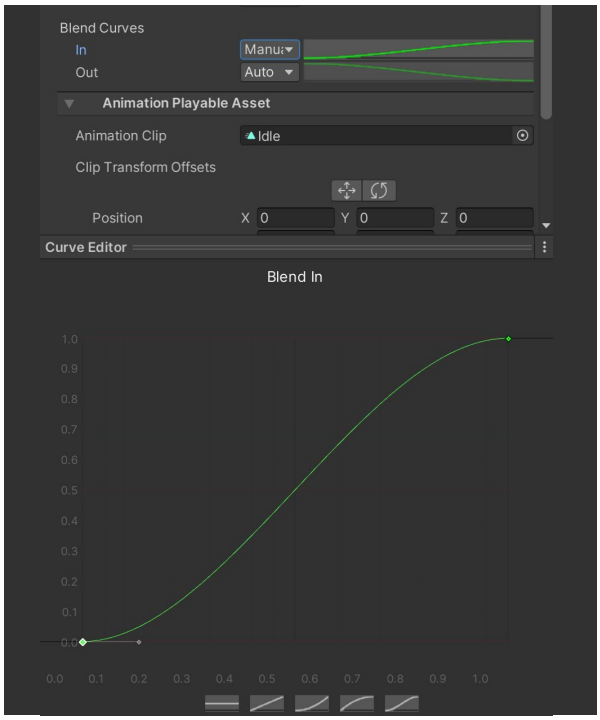


페이드인/페이드아웃 지속 시간 조정

그런 다음 두 번째 애니메이션을 추가하여 첫 번째와 겹치도록 드래그하면 블렌드 오버랩을 형성하여 함께 블렌딩할 수 있습니다.



타임라인에서 두 애니메이션을 함께 블렌딩

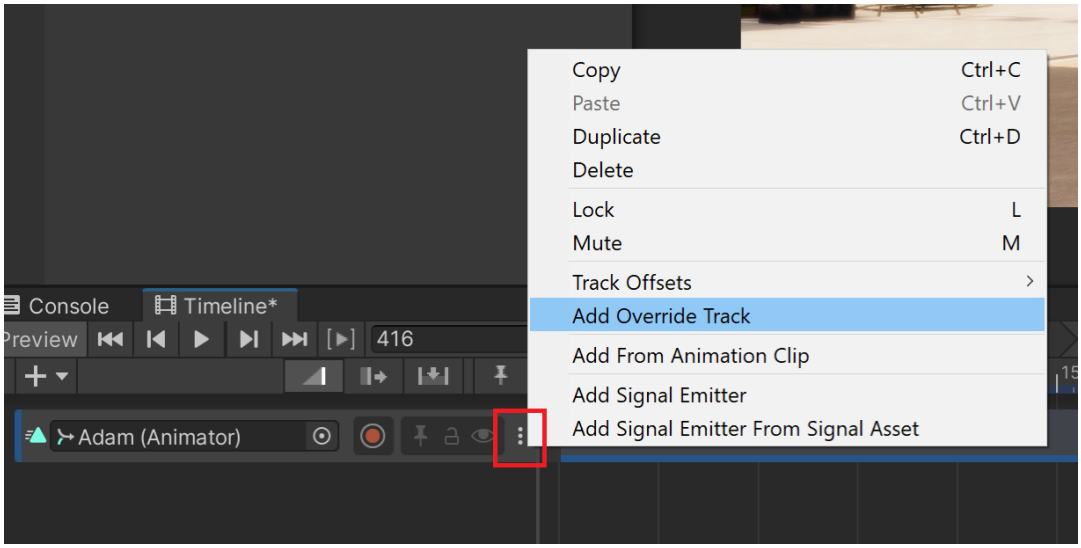


Blend In 커브 조정

Speed Multiplier 설정을 사용하여 클립의 재생 속도를 변경할 수도 있습니다. 1 이외의 속도로 설정하면 애니메이션 트랙 위에 점선으로 표시됩니다. 블렌드 커브를 Manual로 설정하고 커브를 클릭하면 두 클립 간의 이징 방식을 변경할 수 있습니다. 하단의 프리셋을 사용할 수도 있고 베지어 핸들을 조정하여 직접 커브를 생성할 수도 있습니다.

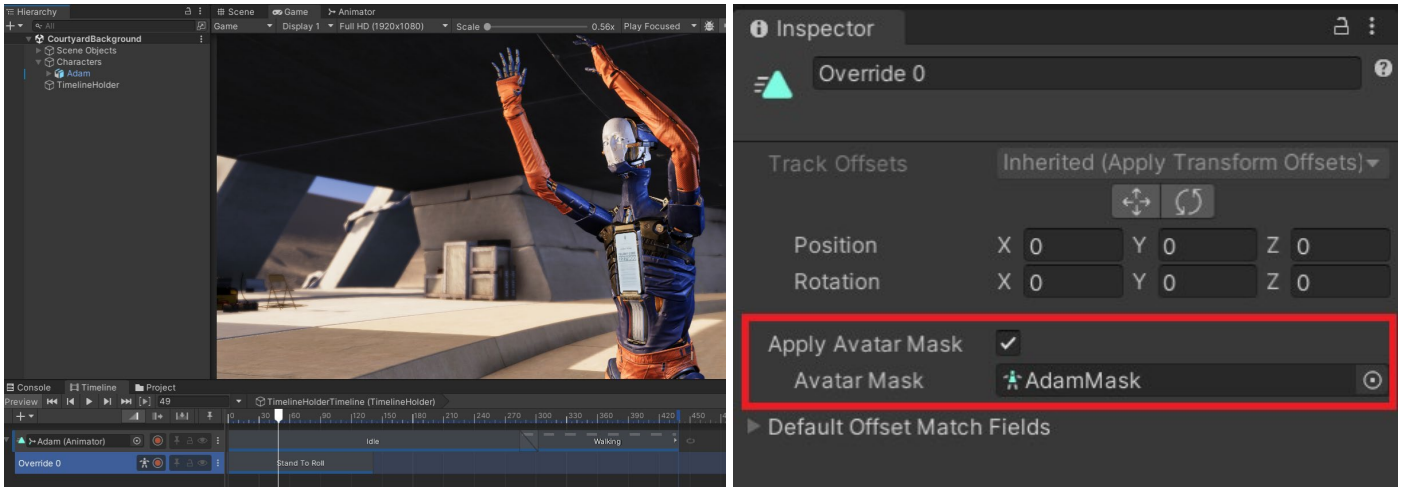
트랙 오버라이드 추가

아바타 마스크를 사용하여 Animator 창과 마찬가지로 둘 이상의 애니메이션을 병합할 수 있습니다. 점 3개의 트랙 옵션 아이콘을 클릭하고 **Add override track**을 선택합니다.



애니메이션 트랙에 오버라이드 트랙 추가

아바타 마스크를 추가할 트랙을 클릭합니다. 이 트랙에 배치하는 애니메이션으로 기본 트랙의 뼈대를 오버라이드하여 흥미롭고 독특한 애니메이션을 제작할 수 있습니다.

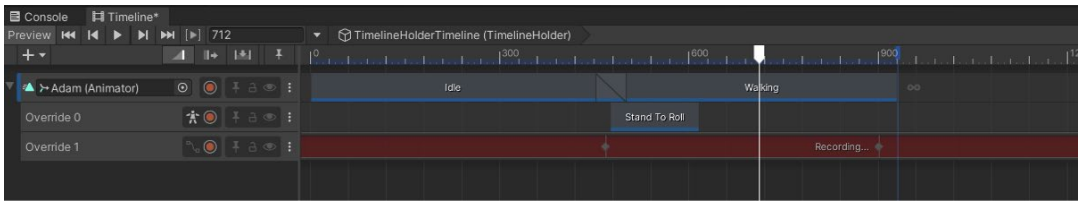


오버라이드 트랙에 아바타 마스크를 사용하여 두 애니메이션을 병합할 수 있습니다.

루트 모션을 사용하지 않는 고정 애니메이션의 경우 오버라이드 트랙으로 캐릭터의 위치, 회전, 스케일을 제어할 수 있습니다.

애니메이션 트랙에 원하는 만큼 오버라이드 트랙을 추가할 수 있지만, 두 오버라이드 트랙이 같은 뼈대를 제어하는 경우 목록 가장 아래의 오버라이드 트랙이 상위의 모든 트랙을 오버라이드합니다.

새 오버라이드 트랙에서 **녹화** 버튼을 클릭하고 캐릭터 위치를 조정하며 무한 클립 에 키프레임을 추가합니다. 녹화 도중에는 트랙이 빨간색으로 표시됩니다.



오버라이드 트랙을 사용하여 캐릭터의 움직임 기록

오른쪽 클릭하고 **Convert to Clip Track**을 선택하여 키프레임을 애니메이션 클립으로 전환합니다. 이렇게 하면 속도, 반복, 블렌드 스타일 등 애니메이션 클립의 모든 옵션을 사용할 수 있습니다.

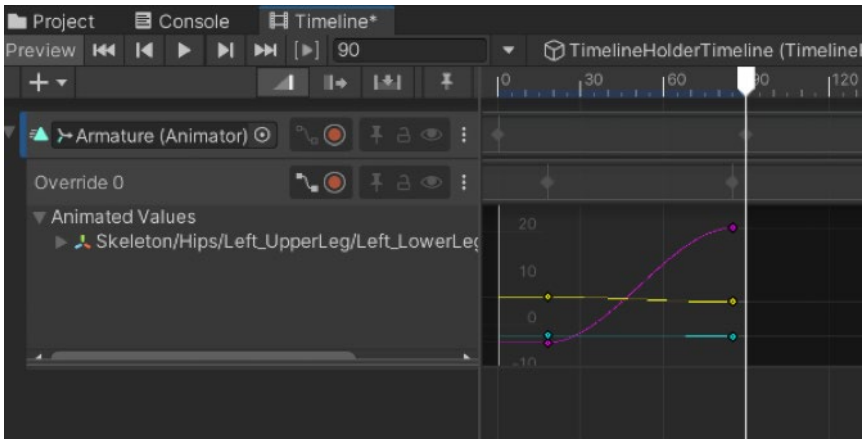
애니메이션 클립을 더블 클릭하여 Animation 창에서 키프레임을 엽니다. 여기에서 이 전자책 앞부분에서 다룬 것과 같이 키프레임을 조정하고 이벤트를 추가할 수 있습니다.

Ctrl+D를 누르거나 **Edit > Duplicate**를 통해 애니메이션 클립을 복제할 수 있습니다. 첫 번째 클립에서 두 번째 클립으로 전환할 때 캐릭터 위치가 원점으로 돌아가는 것을 볼 수 있습니다. 이를 해결하려면 두 번째 클립을 오른쪽 클릭하고 **Match Offsets to Previous Clip**을 선택합니다. 이제 루트 모션을 사용하는 애니메이션처럼 작동하며, 캐릭터가 원점으로 돌아가지 않고 이전 클립의 마지막 위치에서 계속 동작합니다. 현재 클립 다음에 또 다른 클립이 있다면 다음 클립에 오프셋을 맞출 수도 있습니다.

키프레임 설정

타임라인을 사용하여 이 창에서 바로 애니메이션을 만들 수도 있습니다. 제네릭 애니메이션 유형의 경우 순운동학(FK)을 사용하여 각 뼈대의 회전을 조정할 수 있습니다. 뼈대의 위치를 조정하고 섬세하게 다듬기 위해 오버라이드를 사용할 수도 있습니다.

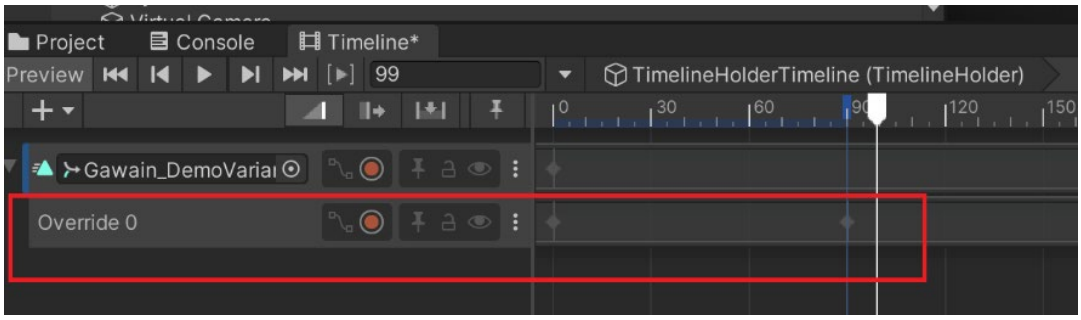
애니메이션 커스터마이징이 가능한 오버라이드를 사용하면 기본 애니메이션도 키프레임을 설정하여 조정할 수 있습니다. 여기에서 커브에 액세스하고 키프레임 간 이징을 조정할 수도 있습니다.



기존 애니메이션 및 커브 조정을 위한 제네릭 릭 키프레임 오버라이드

휴머노이드 애니메이션 유형은 순운동학(FK) 키프레임 설정을 통해 애니메이션화할 수 없습니다. 대신 Animation Rigging 패키지의 역운동학(IK) 릭을 사용하여 타겟의 위치를 애니메이션화할 수 있습니다. 이 전자책의 애니메이션 리깅 섹션을 참고하세요.

씬에서 릭 업데이트를 확인하려면 먼저 Rig Builder 컴포넌트를 가지고 있는 부모 게임 오브젝트의 초기 트랜스폼 키프레임을 설정해야 합니다. 오버라이드 트랙을 추가하면서 타겟의 위치와 회전을 조정하고 씬 뷰에서 실시간으로 업데이트되는 것을 확인할 수 있습니다.

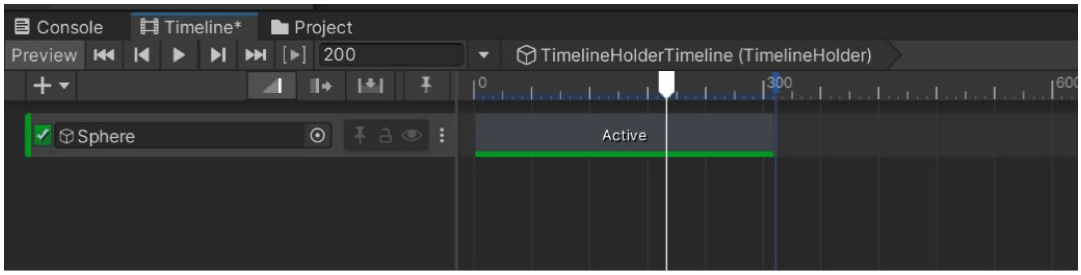


조정된 IK 릭을 실시간으로 애니메이션화

트랙 유형

활성화 트랙

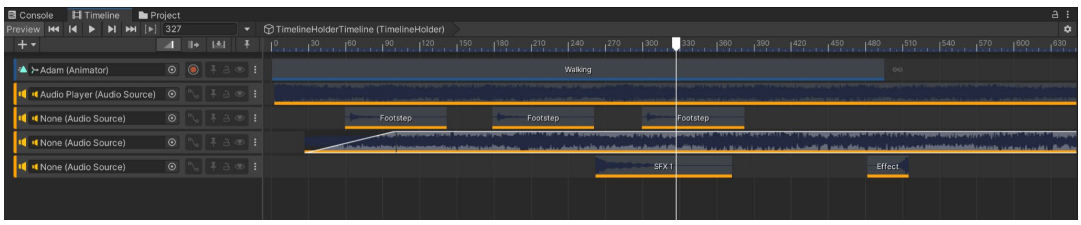
활성화 트랙을 사용하면 애니메이션 블록이 재생되는 시간 동안 오브젝트를 켤 수 있습니다.



블록이 지속되는 동안 오브젝트를 켜는 활성 블록

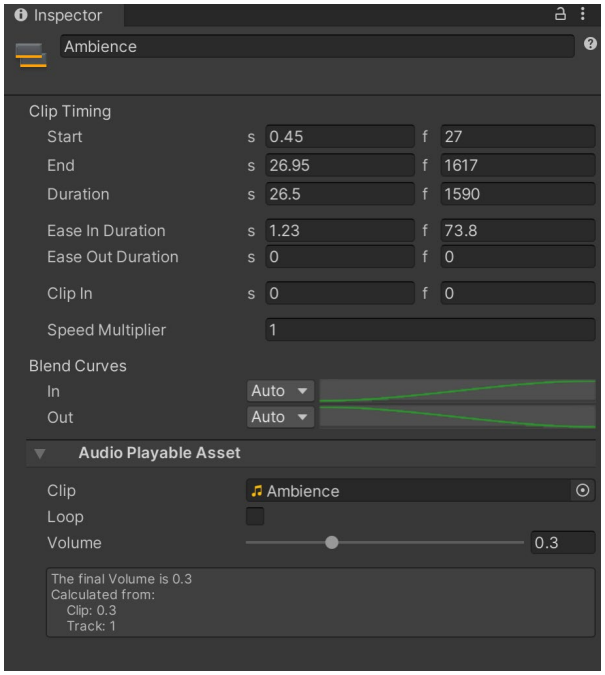
오디오 트랙

오디오 트랙을 사용하면 애니메이션의 특정 시점에 사운드 파일과 음악을 배치할 수 있습니다. 오디오 트랙을 여러 개 사용하여 혼합해 복잡한 음악으로 만들 수도 있습니다.



여러 오디오 트랙을 사용하여 시퀀스에 풍부한 사운드 트랙을 만들 수 있습니다.

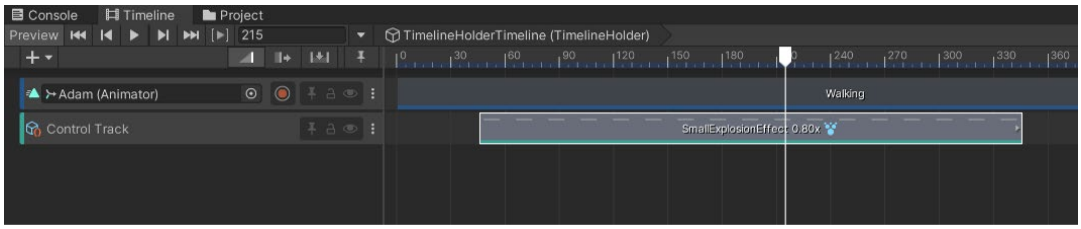
오디오 소스 슬롯이 비어 있으면 사운드와 음악은 2D 공간에서 재생됩니다. Audio Source 컴포넌트가 있는 오브젝트를 이 슬롯에 배치하면 3D 월드의 특정 부분에서 3D 사운드를 재생하도록 설정하여 서라운드 사운드 음악을 구현할 수 있습니다.



사운드 페이드인/페이드아웃 및 볼륨 조절

Ease In Duration과 **Ease Out Duration** 설정을 사용하면 트랙을 페이드인하거나 페이드아웃할 수 있습니다. 트랙에 표시되는 대각선이 페이드인 시간을 나타냅니다. 인스펙터에서 볼륨을 조절할 수도 있습니다.

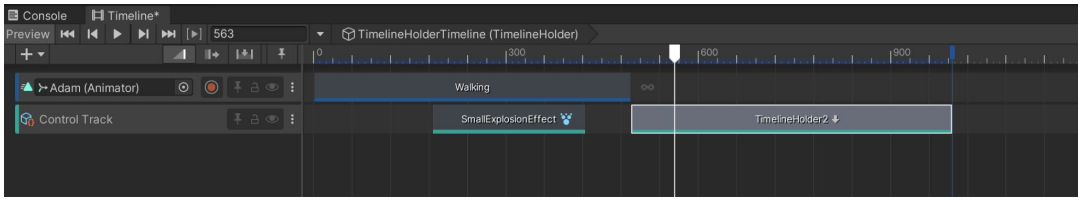
컨트롤 트랙



컨트롤 트랙을 사용하여 파티클 효과 제어

컨트롤 트랙을 사용하면 씬에서 파티클 효과를 활성화할 수 있습니다. 예를 들어 애니메이션의 특정 시점에 폭발이 일어나도록 타이밍을 지정할 수 있죠. 파티클 트랙이 종료된 후에 비활성화되거나, 활성화된 설정으로 계속 반복하거나, Revert 설정을 사용하여 휴면 상태로 돌아가거나, 나중에 다시 활성화되기를 기다리는 등 어떻게 작동할지 선택할 수 있습니다.

컨트롤 트랙을 사용하면 현재 타임라인에서 다른 타임라인을 오케스트레이션할 수 있습니다. 특정 시점에 여러 개의 타임라인 시퀀스를 시작하려는 경우 TimelineHolder 게임 오브젝트를 컨트롤 트랙에 드래그하여 적절한 시간에 활성화하면 됩니다.



컨트롤 트랙을 사용하여 여러 타임라인 제어

플레이어블 트랙

플레이어블 트랙을 사용하면 커스텀 스크립트를 만들고 이 트랙에 추가하여 시퀀스 도중 오브젝트와 캐릭터를 조작할 수 있습니다. 더 자세한 내용은 이 Unity 블로그 게시물에서 살펴보세요.

시그널 트랙

시그널 트랙은 Animation 창의 이벤트와 매우 유사하게 작동합니다. 게임 오브젝트의 컴포넌트와 오브젝트에 연결된 스크립트의 커스텀 Public 함수에 액세스하는 시그널 이벤트를 설정할 수 있습니다.

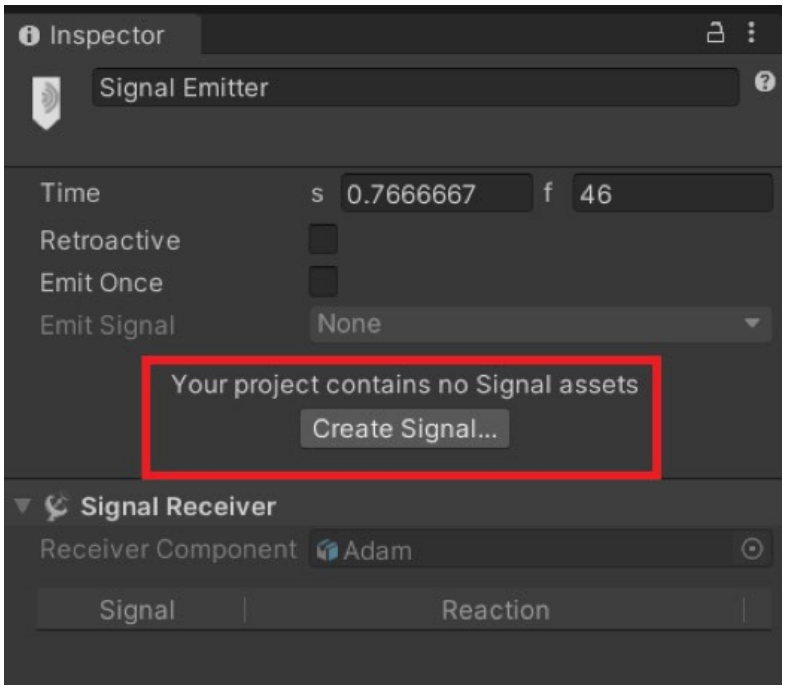
캐릭터의 걸음걸이와 일치하는 발소리를 생성하려면 각각 왼쪽 발과 오른쪽 발에 해당하는 Public 함수 2개가 포함된 스크립트를 만들면 됩니다.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

Unity Script | 0 references
public class Footsteps : MonoBehaviour
{
    private AudioSource m_AudioSource;
    public AudioClip leftFooptStep;
    public AudioClip RightFooptStep;

    Unity Message | 0 references
    private void Start()
    {
        m_AudioSource = GetComponent<AudioSource>();
    }

    0 references
    public void PlayLeftFootSound()
    {
        m_AudioSource.clip = leftFooptStep;
        m_AudioSource.Play();
    }
    0 references
    public void PlayRightFootSound()
    {
        m_AudioSource.clip = RightFooptStep;
        m_AudioSource.Play();
    }
}
```

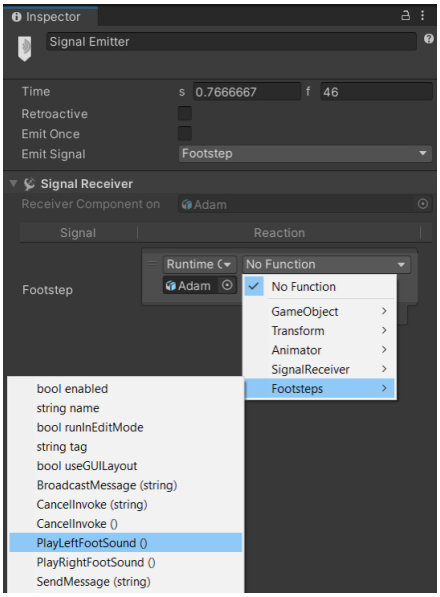
발소리를 재생하는 스크립트

사운드를 재생할 Audio Source 컴포넌트를 스크립트와 함께 캐릭터에 연결합니다.

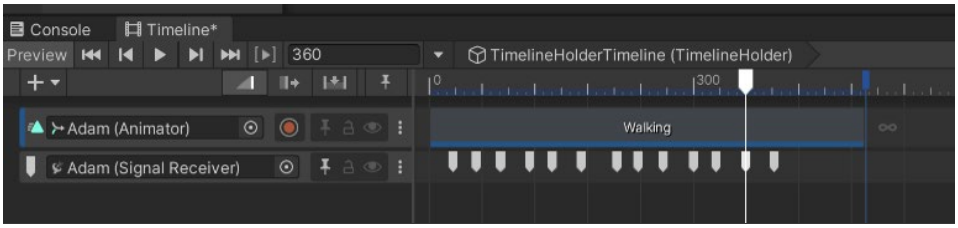
이제 타임라인에 시그널 트랙을 추가합니다. 첫 번째 발자국 시점에서 오른쪽 클릭하고 Signal Emitter를 추가합니다. 시그널 에셋이 아직 없으므로 노란색 경고가 표시될 것입니다. 이 에셋은 인스펙터에서 생성하면 됩니다.



새로운 시그널 에셋 생성



드롭다운에서 캐릭터에 연결된 모든 컴포넌트에 액세스하고 여러 작업을 수행할 수 있습니다. 스크립트를 선택하면 PlayLeftFootSound 및 PlayRightFootSound 함수를 찾아 좌우 발소리에 여러 시그널을 추가할 수 있습니다.



시그널 에셋에서 커스텀 함수 선택

Cinemachine 트랙

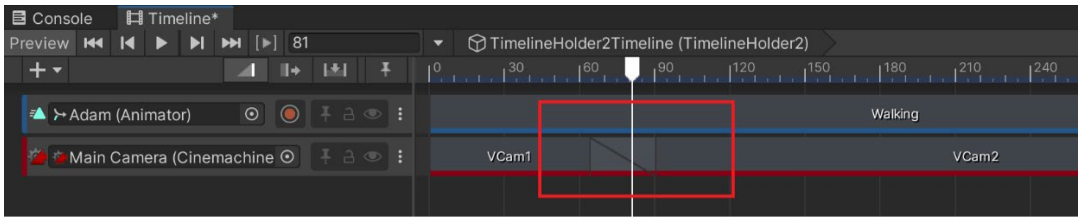
Cinemachine은 복잡한 시네마틱 카메라 설정을 효율적으로 생성합니다. Cinemachine 트랙을 사용하면 Cinemachine에 사용되는 가상 카메라를 제어하여 샷의 타이밍과 구도를 조정할 수 있습니다.

메인 카메라를 빈 슬롯으로 드래그하면 해당 메인 카메라에 Cinemachine Brain 컴포넌트가 생성됩니다. 이 컴포넌트는 메인 카메라가 가상 카메라 설정을 따르도록 하는 데 사용됩니다.

가상 카메라를 몇 개 만들어 프레이밍하면 가상 카메라가 이 트랙에 추가됩니다. 한 카메라에서 다음 카메라로 넘어가면 둘 사이에 단절이 발생합니다.

두 카메라 샷 사이가 단절되는 Cinemachine

한 카메라 블록을 다른 카메라 블록으로 드래그하면 크로스오버 블렌드가 적용되어 카메라가 첫 번째 샷과 다음 샷 사이를 이동합니다.



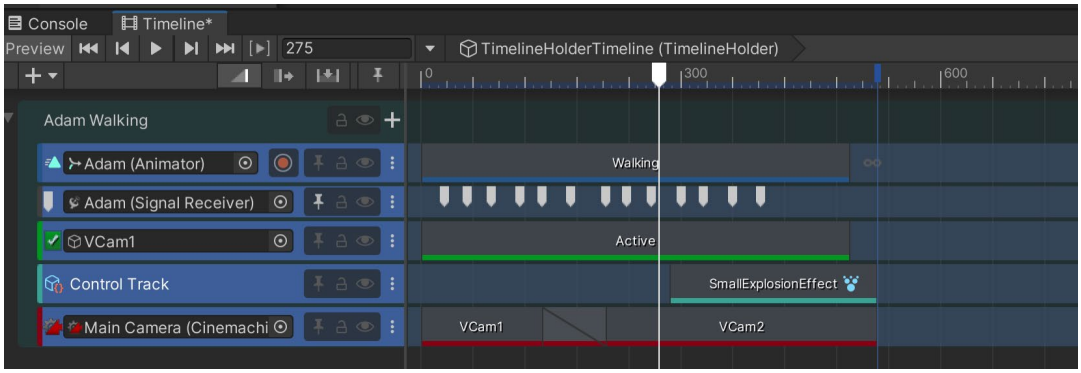
두 카메라 샷 사이가 블렌딩되는 Cinemachine

위 스크린샷에서는 Cinemachine 버전 2가 사용되었습니다. Cinemachine 버전 3의 새로운 기능에 대한 자세한 내용은 이 블로그 글을 참조하세요.

트랙 그룹

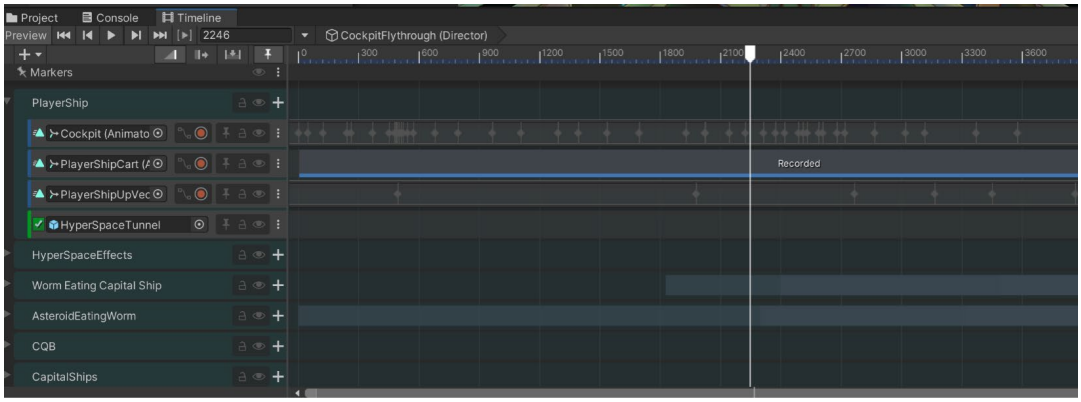
트랙을 추가할수록 타임라인을 탐색하기가 어려워질 수 있습니다.

트랙 그룹을 사용하면 특정 트랙을 함께 묶어 정리할 수 있죠.



트랙 그룹으로 한데 묶인 트랙

그런 다음 트랙 그룹을 최소화해 타임라인에 더 많은 공간을 확보하면 더 용이하게 탐색할 수 있게 됩니다.



최소화된 트랙 그룹

시퀀스

Sequences 패키지를 사용하면 타임라인에서 작업할 때 큰 프로젝트를 더 쉽게 처리할 수 있습니다. 재생 중에 특정 시점에 씬을 동적으로 로드할 수 있으므로 더 쉽게 씬을 작업할 수 있는 것이죠.

Sequences 패키지 사용법에 대한 단계별 개요는 이 Unity Learn 튜토리얼을 참조하세요.

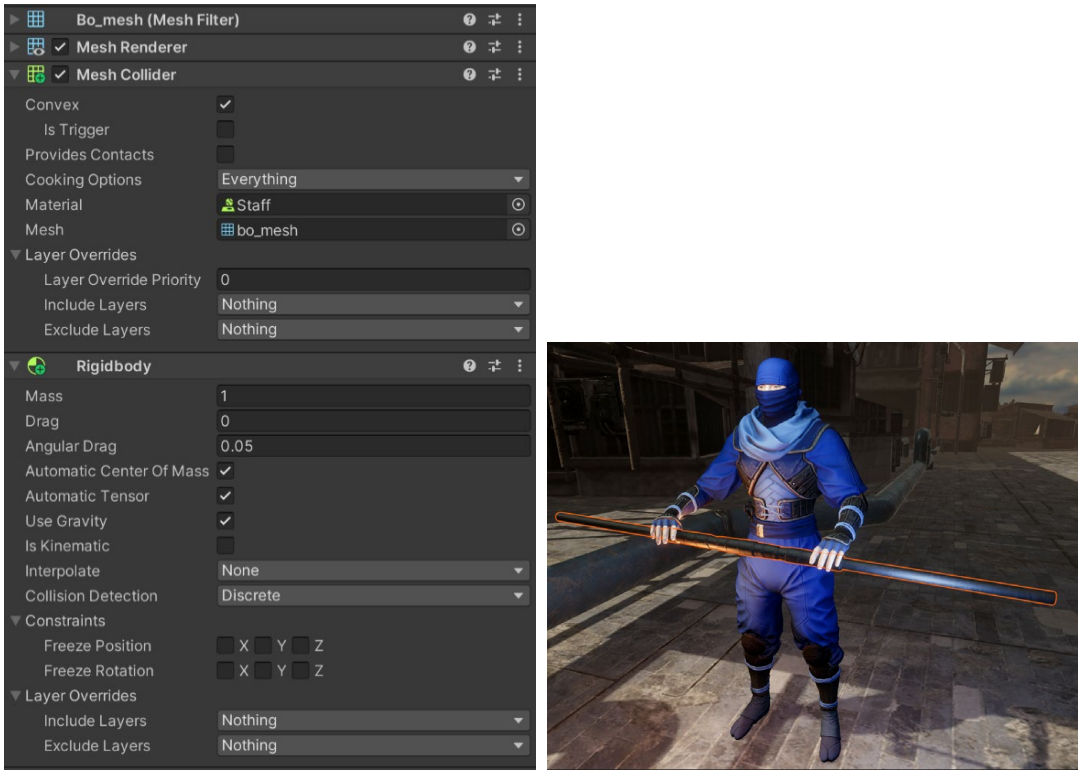
고급 시뮬레이션

Unity의 빌트인 3D 물리 엔진은 물리 이벤트를 실시간으로 시뮬레이션합니다. 중력에 반응하는 오브젝트와 모발, 모피, 천, 망토, 커튼, 깃발 같은 오브젝트의 사실적인 움직임을 시뮬레이션하는 데 사용할 수 있습니다.

리지드바디 물리

고체 물리를 추가하려면 오브젝트에 Rigidbody 컴포넌트를 추가하고 Mesh Collider 컴포넌트를 추가하여 바닥을 뚫고 내려가지 않도록 합니다.

고체 물리는 가상 환경에서 오브젝트 간의 물리적 상호 작용을 사실적으로 시뮬레이션하는 것을 의미합니다. 질량, 중력, 마찰, 충돌 같은 프로퍼티를 시뮬레이션하여 현실 세계를 모방하는 방식으로 오브젝트의 움직임을 보여 줍니다.



물리 효과를 내기 위해 Rigidbody 컴포넌트가 연결된 지팡이 게임 오브젝트

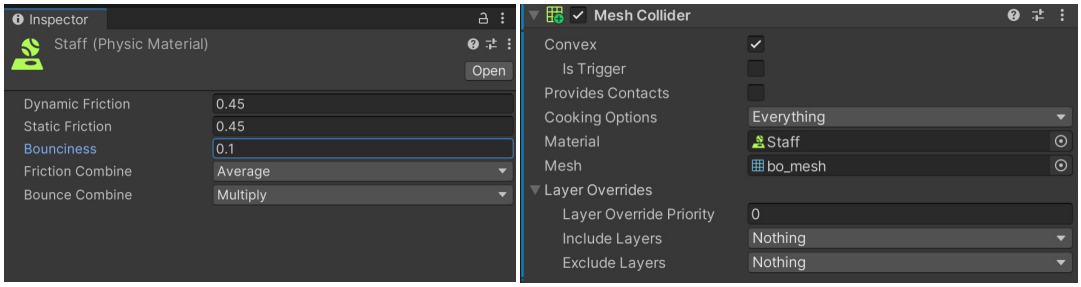
일정한 질량과 항력의 Rigidbody 컴포넌트를 설정하면 현실에서 오브젝트가 낙하할 때 공기와 상호 작용하는 방식을 시뮬레이션할 수 있습니다. 특정 레이어를 포함하거나 제외하여, 오브젝트가 설정된 레이어에 있는 다른 오브젝트에만 충돌하도록 설정할 수 있습니다. **Freeze Position** 및 **Freeze Rotation** 설정을 사용하여 시뮬레이션을 더 세밀하게 제어할 수도 있습니다.

IsKinematic 프로퍼티를 선택하지 않은 상태로 두면 오브젝트에서 중력을 사용합니다. IsKinematic을 선택하면 오브젝트에서 중력을 사용하지 않지만 충돌 같은 물리 기반 상호 작용은 여전히 처리할 수 있습니다.

콜라이더는 기본적으로 물리 콜라이더이며 오브젝트가 다른 오브젝트를 통과하는 상황을 방지합니다. 그러나 **Is Trigger** 옵션을 선택하면 물리적 측면을 비활성화하는 트리거 콜라이더로 해당 콜라이더를 설정하여 오브젝트가 다른 오브젝트를 통과하지만 다른 오브젝트와 교차 시에는 여전히 트리거를 반환하도록 구현할 수 있습니다. 이 옵션은 코딩에 유용할 수 있습니다.

메시 콜라이더에 물리 머티리얼 에셋을 추가하여 현실 오브젝트를 시뮬레이션하는 식으로 오브젝트가 동작하도록 설정할 수 있습니다. 예를 들어 탄력 있는 고무공의 경우, Collider 컴포넌트의 **Material** 필드에 탄력성 물리 머티리얼을 추가해야 합니다.

새 물리 머티리얼을 생성하려면 프로젝트 창에서 오른쪽 클릭하고 **Create > Physic Material**을 선택한 후 값을 조정하여 원하는 결과를 얻습니다.

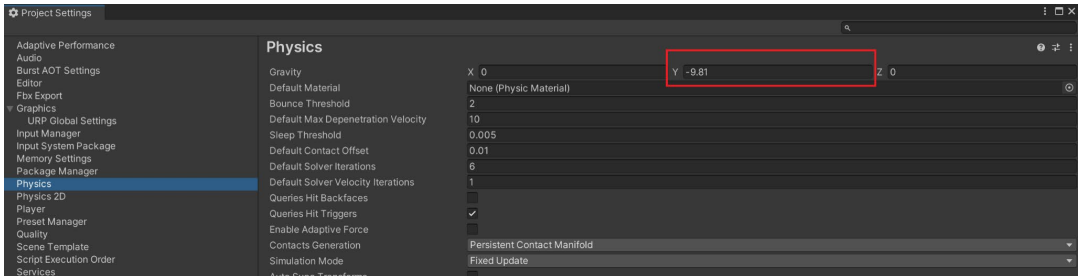


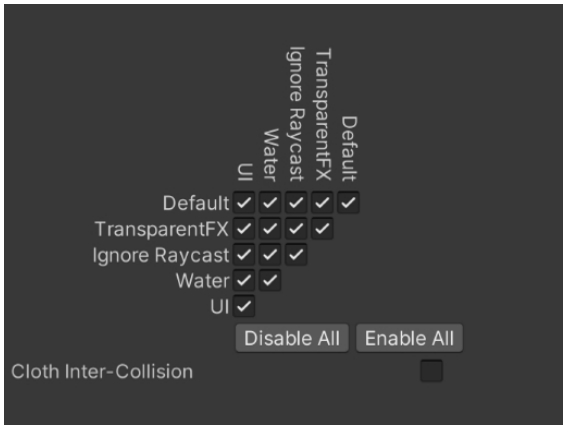
나무 지팡이의 물리 머티리얼을 정의한 후 메시 콜라이더에 할당

머티리얼	Dynamic Friction	Static Friction	Bounciness	Friction Combine	Bounce Combine
탄성체	0.3	0.3	1	Average	Maximum
얼음	0.1	0.1	0	Multiply	Multiply
최대 마찰	1	1	0	Maximum	Average
금속	0.15	0.15	0	Minimum	Average
고무	1	1	0.5	Maximum	Average
나무	0.45	0.45	0	Average	Average
마찰 없음	0	0	0	Multiply	Average

다양한 유형의 오브젝트에 유용한 물리 프리셋

Edit > Project Settings로 이동하여 Physics 설정을 조정할 수 있습니다. 여기서 중력량을 변경할 수 있습니다. 기본적으로는 현실의 중력량인 -9.81로 설정되어 있지만 값을 변경하여 우주를 유영하는 캐릭터를 시뮬레이션할 수 있습니다.

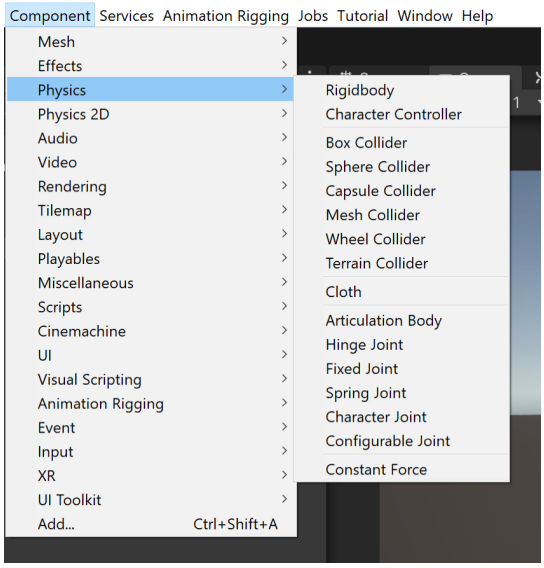




Physics 설정은 Project Settings 창을 통해 액세스합니다.

하단의 레이어 충돌 매트릭스를 사용하여 어떤 레이어가 서로 상호 작용할 수 있는지 정의합니다.

Component > Physics에서 다양한 물리 콜라이더, 조인트, 힘을 추가할 수 있습니다.



Physics 컴포넌트

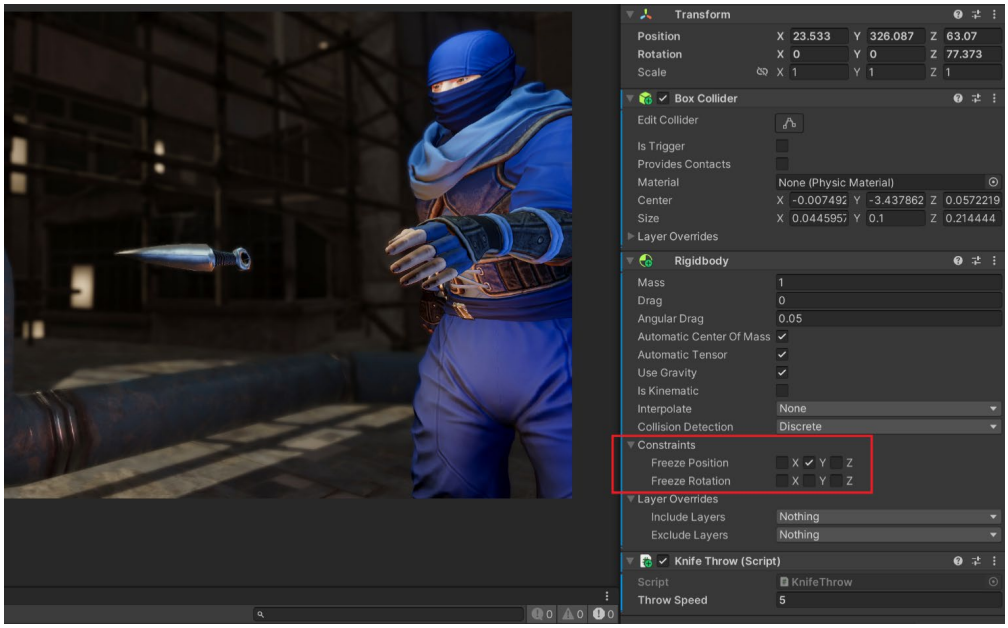
상수 힘(constant force)은 바람 또는 투척 오브젝트를 시뮬레이션하는 경우처럼 물리적 오브젝트에 특정 방향으로 힘을 가하고 싶을 때 유용합니다.

코드를 통해 Rigidbody 컴포넌트에 힘을 추가할 수도 있습니다. 예를 들어 마우스 클릭에 힘을 추가하여 설정된 속도로 칼을 전방으로 움직이는 것입니다.

```
KnifeThrow.cs
Assembly-CSharp                                    KnifeThrow
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

Unity Script | 0 references
public class KnifeThrow : MonoBehaviour
{
    private Rigidbody rb;
    public float throwSpeed = 1.5f;

    // Start is called before the first frame update
    Unity Message | 0 references
    void Start()
    {
        rb = GetComponent<Rigidbody>();
    }

    // Update is called once per frame
    Unity Message | 0 references
    void Update()
    {
        if (Input.GetMouseButtonDown(0))
        {
            rb.isKinematic = false;
            rb.AddForce(Vector3.forward * throwSpeed * Time.deltaTime, ForceMode.Impulse);
        }
    }
}
```

플레이어가 클릭 시 오브젝트에 힘을 추가하는 코드

이때 Rigidbody 컴포넌트는 Y 이동 축에 고정되어 있으므로 칼은 바닥에 떨어지지 않고 코드에서 설정한 힘의 속도로 전진합니다.



칼 투척 및 Y축 위치 제한

Rigidbody 컴포넌트를 사용하여 캐릭터의 움직임을 제어할 수 있습니다. 게임 월드의 물리 오브젝트에 반응하는 사실적인 모션을 제작하는 데 유용한 기능입니다.

이를 구현하려면 코드를 통해 Rigidbody 컴포넌트에 액세스하고 MovePosition 또는 MoveRotation 커맨드를 사용합니다.

```
public class BasicMoveNoRootMotion : MonoBehaviour
{
    private Animator animator;
    private Rigidbody rb;
    private Vector3 rotateY;
    //public settings
    public float walkSpeed = 2.5f;
    public float rotateSpeed = 150.0f;

    Unity Message | 0 references
    void Start()
    {
        animator = GetComponent<Animator>();
        rb = GetComponent<Rigidbody>();
    }

    Unity Message | 0 references
    void Update()
    {
        rb.MovePosition(rb.position + Input.GetAxis("Vertical") * Time.deltaTime * walkSpeed * transform.forward);
        if (Input.GetAxis("Vertical") != 0)
        {
            if (Input.GetAxis("Horizontal") > 0)
            {
                rotateY = new Vector3(0, rotateSpeed * Time.deltaTime, 0);
            }
            if (Input.GetAxis("Horizontal") < 0)
            {
                rotateY = new Vector3(0, -rotateSpeed * Time.deltaTime, 0);
            }
            rb.MoveRotation(rb.rotation * Quaternion.Euler(rotateY));
        }
        animator.SetFloat("Vertical", Input.GetAxis("Vertical"));
    }
}
```

Rigidbody 이동을 사용하여 캐릭터의 움직임 제어

점프하려면 리지드바디에 힘을 추가합니다.

```
    void Update()
    {
        if (Input.GetButtonDown("Jump"))
        {
            animator.SetTrigger("Jump");
            rb.AddForce(Vector3.up * jumpForce, ForceMode.Impulse);
        }
    }
```

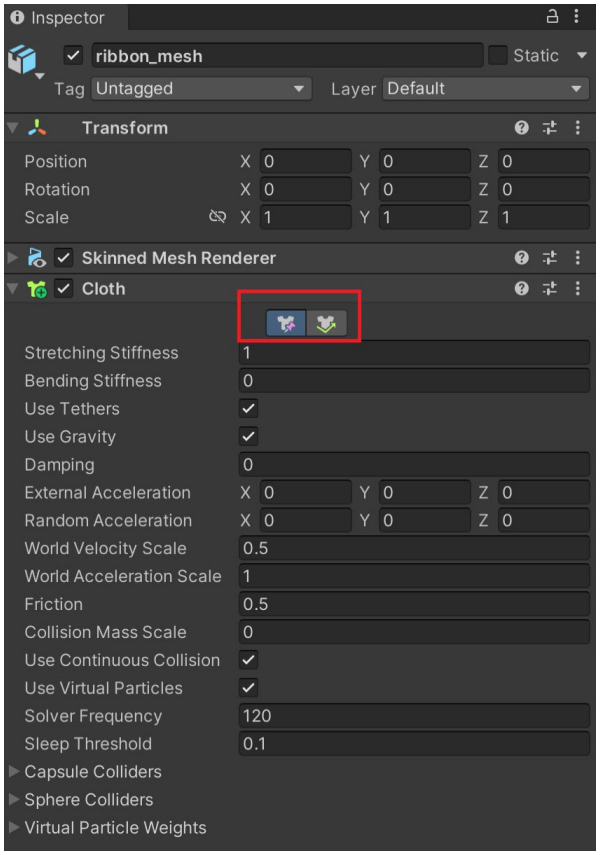
점프하기 위해 Rigidbody 컴포넌트에 힘 추가

천 물리

천 물리는 천의 움직임을 시뮬레이션하며, 높은 수준으로 세분화된 모든 메시에 적용할 수 있습니다. 시뮬레이션된 머티리얼을 사실적으로 움직이게 하려면 메시에 충분한 버텍스가 있어야 하지만, 성능 비용이 증가할 정도로 많으면 안 됩니다.

오브젝트에 Cloth 컴포넌트를 추가한 후, 관련 설정을 사용하여 머티리얼 유형을 정의합니다. **Stretching Stiffness**, **Bending Stiffness** 같은 프로퍼티를 조정하면 가죽, 실크, 면 등을 시뮬레이션할 수 있습니다.

천 오브젝트에 제약 설정

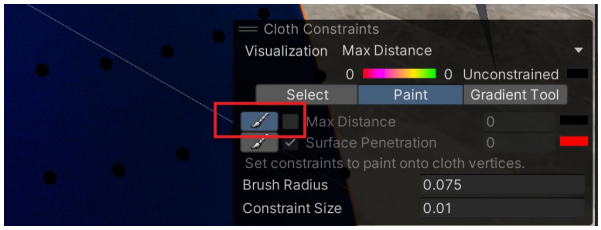


Cloth 프로퍼티와 천 제약 버튼

천은 원점에서 제한하지 않으면 바닥에 떨어집니다. 빨간색 핀이 있는 티셔츠 아이콘(왼쪽 이미지 참조)을 클릭하면 Cloth Constraints 메뉴가 열립니다. **Max Distance**를 0으로 설정하고 천에서 고정할 지점을 빨간색 점으로 칠합니다. 검은색 점은 제약이 적용되지 않는 부분이며 천 시뮬레이션 중에 움직이게 됩니다.

점 사이로 천이 이동하는 거리를 특정 값으로 정의

버텍스 점이 서로 멀어질 수 있는 최대 거리에 특정 수치를 설정할 수도 있습니다. 이 방식은 스크린샷에 나와 있는 닌자 캐릭터의 가죽 도복과 같이 무거운 머티리얼을 정의하는 데 유용합니다. 주황색에서 초록색 사이의 색상은 서로 멀어지는 점에 특정 거리를 정의하여 천에 다양한 유형의 움직임을 부여합니다. 해당 색상은 그레디언트 툴로 지정할 수 있습니다.

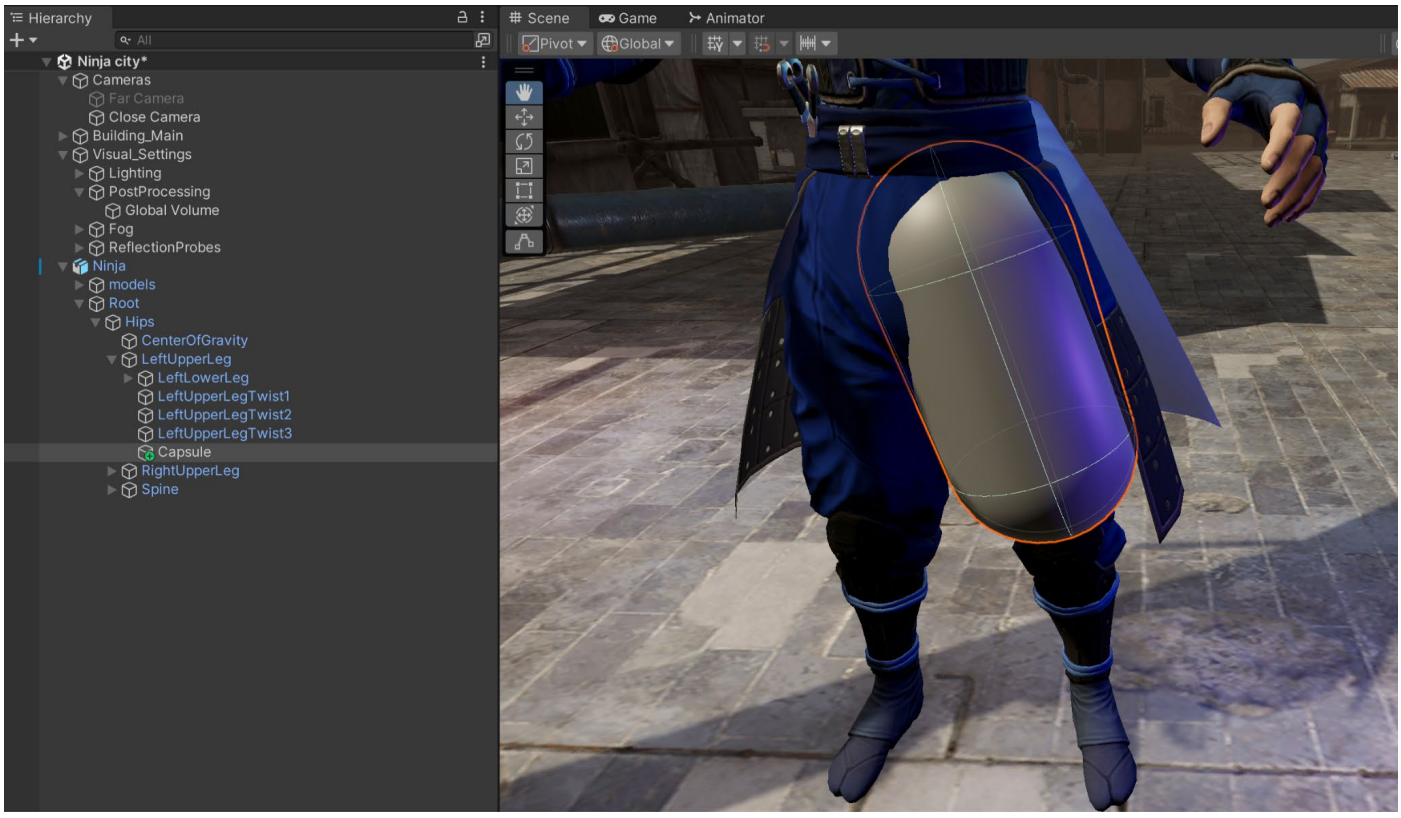


점을 무제약 상태로 되돌림

실수로 Max Distance 체크박스를 선택 해제하여 점 위를 칠하면 이러한 점은 제약이 없는 검은색 점으로 돌아갑니다.

시뮬레이션 시, 천 오브젝트는 캐릭터의 나머지 부분과 교차하므로 천이 캐릭터를 통과하지 못하도록 콜라이더를 배치해야 합니다. 천은 캡슐 콜라이더와 스피어 콜라이더만 지원합니다.

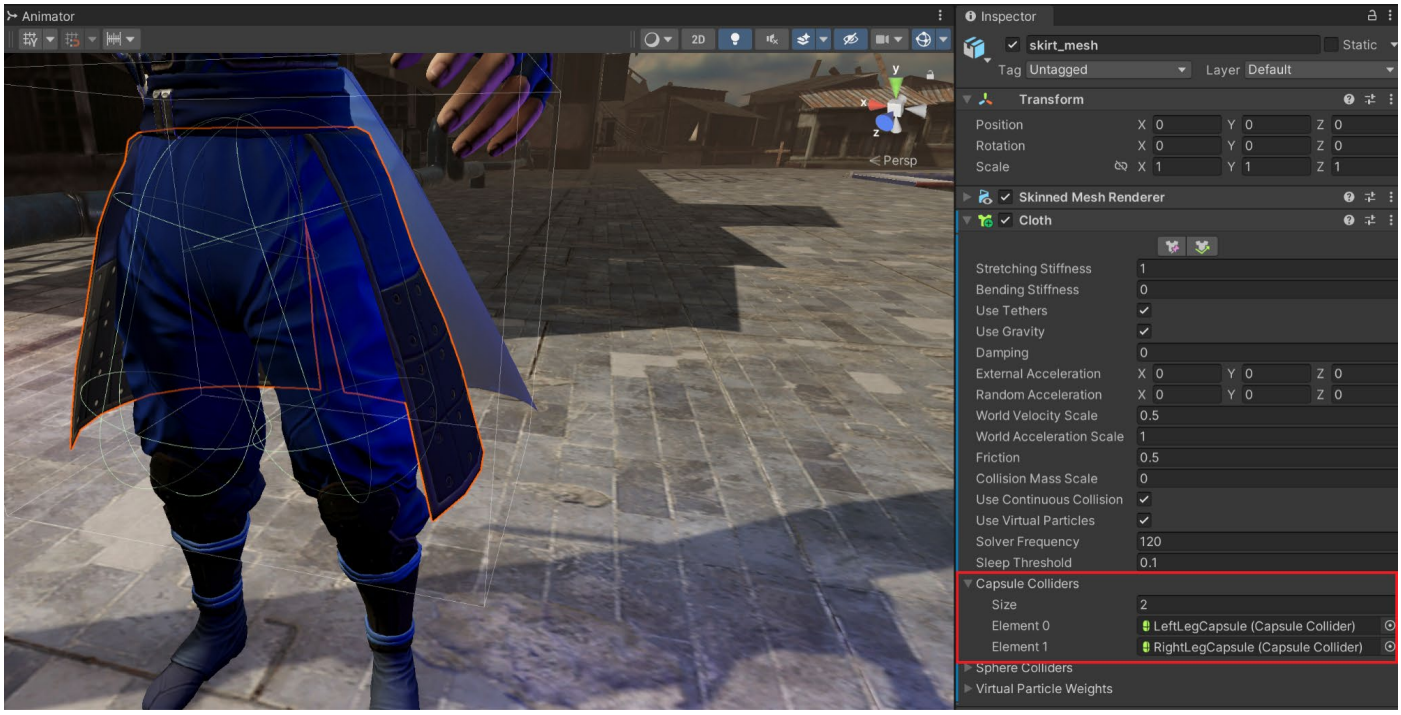
여러 개의 캡슐과 구체를 사용해 정확한 셰이프를 정의하여 복잡한 셰이프를 만든 다음, Cloth 컴포넌트의 캡슐 및 스피어 콜라이더 배열에 이를 추가할 수 있습니다.



뼈대에 콜라이더 추가

계층 구조에서 해당 뼈대에 캡슐 또는 구체를 추가하고 위치를 지정합니다. 콜라이더는 뼈대에 고정되어 캐릭터에게 사실적이고 정확한 움직임을 부여합니다. 콜라이더가 제자리에 배치되면 오른쪽 클릭하고 **Remove Component**를 선택하여 콜라이더에 적용된 Renderer 컴포넌트를 제거합니다.

두 다리에 콜라이더가 적용되면 각 오브젝트의 Cloth 컴포넌트에 콜라이더를 추가합니다.



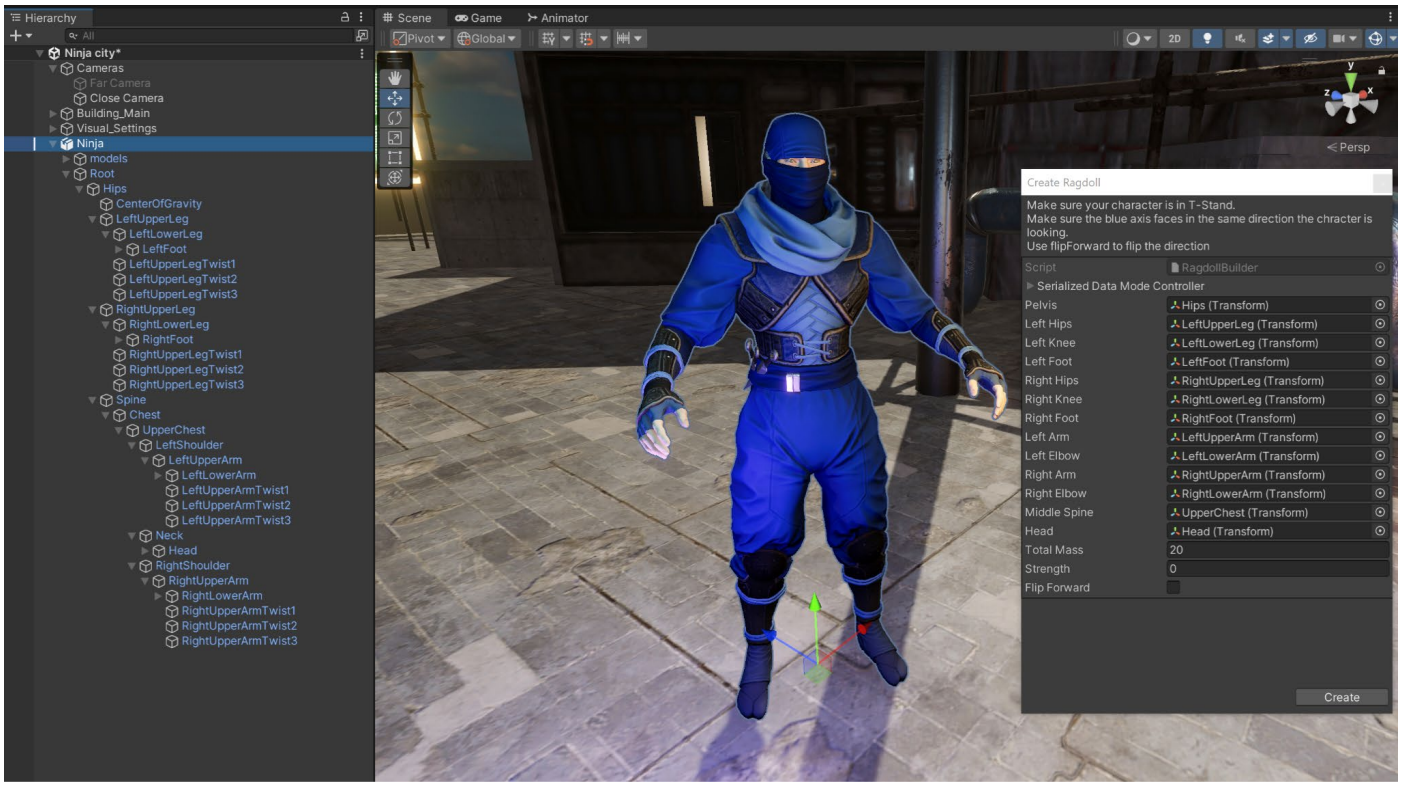
Skinned Mesh Renderer 컴포넌트에 캡슐 추가

사실적인 천 물리 시뮬레이션

래그돌 물리

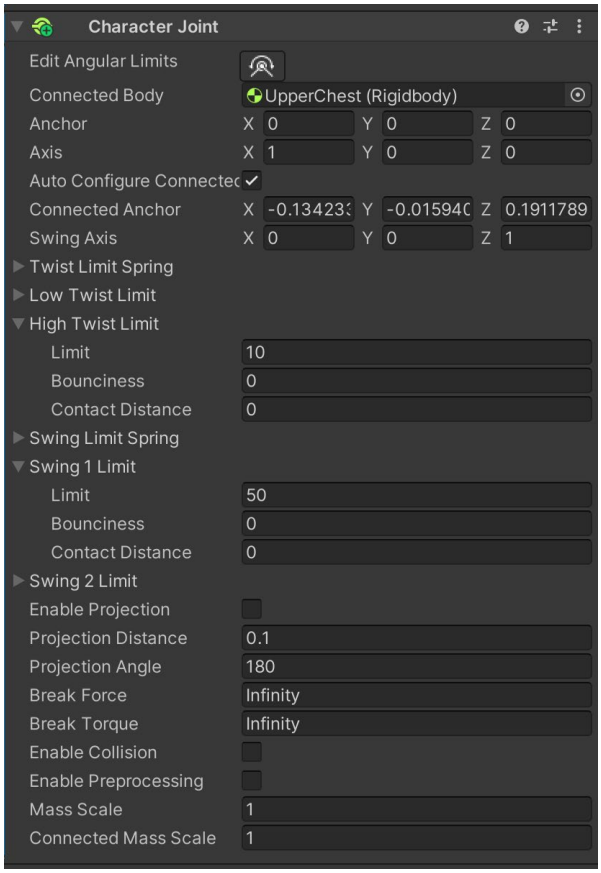
래그돌(Ragdoll) 물리에서는 캐릭터가 넘어지거나 바닥에 쓰러질 때 다리와 팔의 흔들림을 시뮬레이션합니다. Unity는 캐릭터의 골격에 물리 컴포넌트를 추가할 수 있는 빠른 설정 시스템이 있습니다.

캐릭터를 선택하고 **GameObject > 3D Object > Ragdoll**로 이동합니다. 그러면 설정 창이 열립니다. 이때 캐릭터는 T 또는 A 포즈를 취하고 있어야 합니다. 뼈대를 해당 슬롯으로 드래그하고 **Create**를 클릭합니다.



래그돌 설정 창

이제 목록에 있는 각 뼈대에 Rigidbody 컴포넌트가 추가되고, 래그돌 스타일의 물리를 시뮬레이션하는 Character Joint 컴포넌트가 추가됩니다.



래그돌 물리를 처리하는 Character Joint 컴포넌트

Character Joint 컴포넌트를 수정하여 원하는 래그돌 효과를 구현할 수 있습니다. Rigidbody 컴포넌트의 **Mass** 프로퍼티에 설정한 값도 시뮬레이션에 영향을 미칩니다. 이 컴포넌트를 복사하여 목록에 없는 골격의 다른 뼈대에 붙여 넣어 나만의 커스텀 래그돌 설정을 만들 수 있습니다.

재생되는 모든 애니메이션이 래그돌 시뮬레이션을 오버라이드하므로 플레이 모드에서 래그돌 물리를 확인하려면 Animator 컴포넌트를 비활성화해야 합니다.

닌자 캐릭터의 래그돌 시뮬레이션

예를 들어 캐릭터가 총에 맞았을 때 래그돌 효과를 재생하려는 경우에는 스크립트로 Animator 컴포넌트를 비활성화하면 됩니다. 그러나 애니메이션에서 래그돌로 바로 전환하면 캐릭터의 물리가 예측할 수 없는 방식으로 적용되는 이상한 결과가 나타날 수 있습니다.

이 문제를 해결하려면 각 뼈대의 Rigidbody 컴포넌트를 **isKinematic**으로 설정하여 애니메이터가 꺼질 때까지 중력의 영향을 받지 않도록 합니다. 그런 다음 isKinematic을 false로 다시 설정하면 래그돌 효과가 문제 없이 재생되는 것을 확인할 수 있습니다.

```
public class RagdollScript : MonoBehaviour
{
    private Animator anim;
    private Rigidbody[] rigidbodies;

    Unity Message | 0 references
    void Start()
    {
        anim = GetComponent<Animator>();
        rigidbodies = GetComponentsInChildren<Rigidbody>();
        for (int i = 0; i < rigidbodies.Length; i++)
        {
            rigidbodies[i].isKinematic = true;
        }
    }
    0 references
    public void RagdollOn()
    {
        anim.enabled = false;
        for (int i = 0; i < rigidbodies.Length; i++)
        {
            rigidbodies[i].isKinematic = false;
        }
    }
}
```

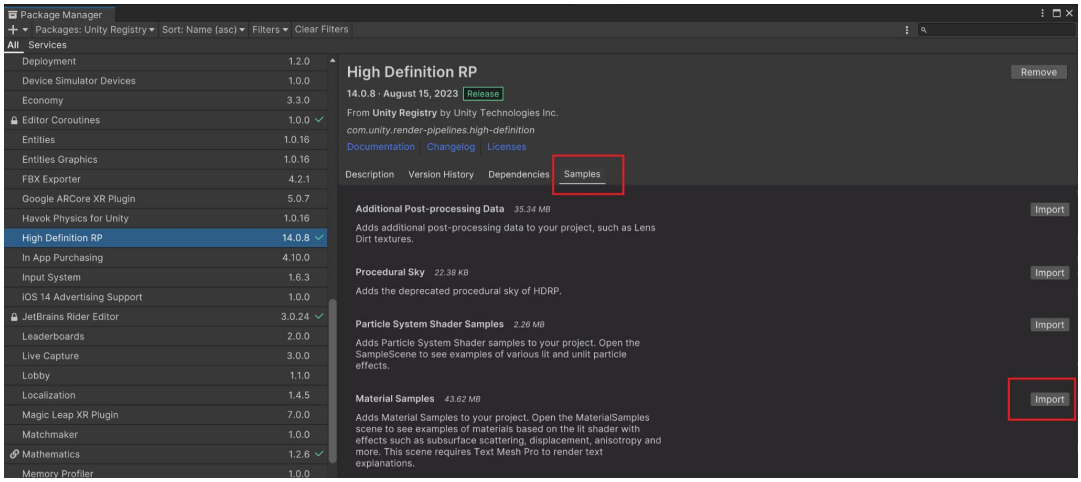
Animator 컴포넌트를 끄면 rigidbodies를 isKinematic으로 전환, 이 작업은 부모 게임 오브젝트에서 이루어짐

모피와 모발

Unity의 HDRP(고해상도 렌더 파이프라인)에서 제공하는 정교한 Shader Graph 기반의 헤어 셰이더를 통해 사실적인 모발과 모피를 구현할 수 있습니다. 라인 기반 렌더러를 사용하여 모발 효과를 직접 구현할 수도 있습니다. 유니티의 사자 데모에서 실제 활용 사례를 확인하세요.

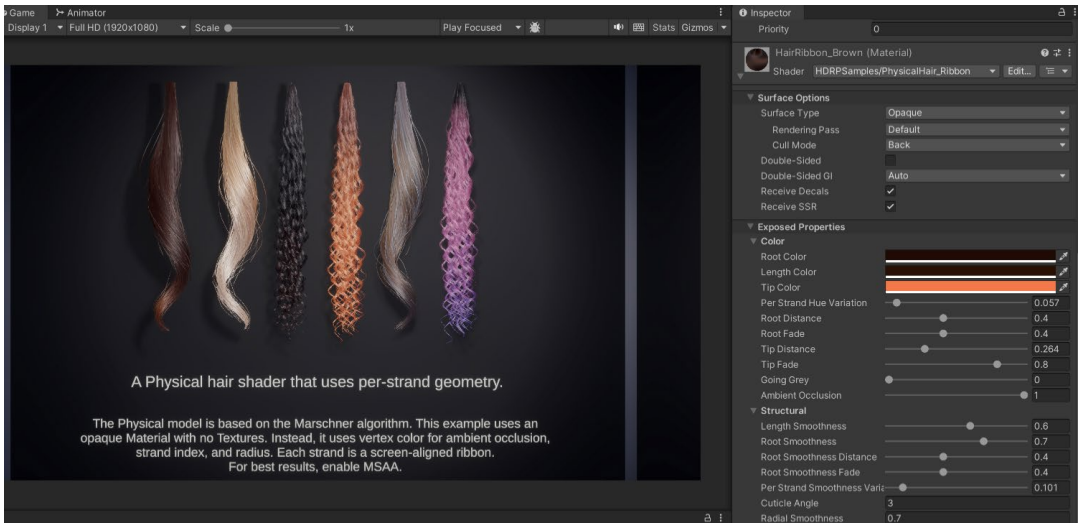
이 사자 데모에서는 Unity 에디터에서 Wētā Digital, SpeedTree, Ziva, SyncSketch를 비롯한 아티스트 툴로 제작한 실시간 콘텐츠를 선보입니다.

패키지 관리자의 High Definition RP 섹션에서 Materials Samples 폴더를 설치하여 헤어 셰이더의 샘플 씬을 살펴볼 수 있습니다.



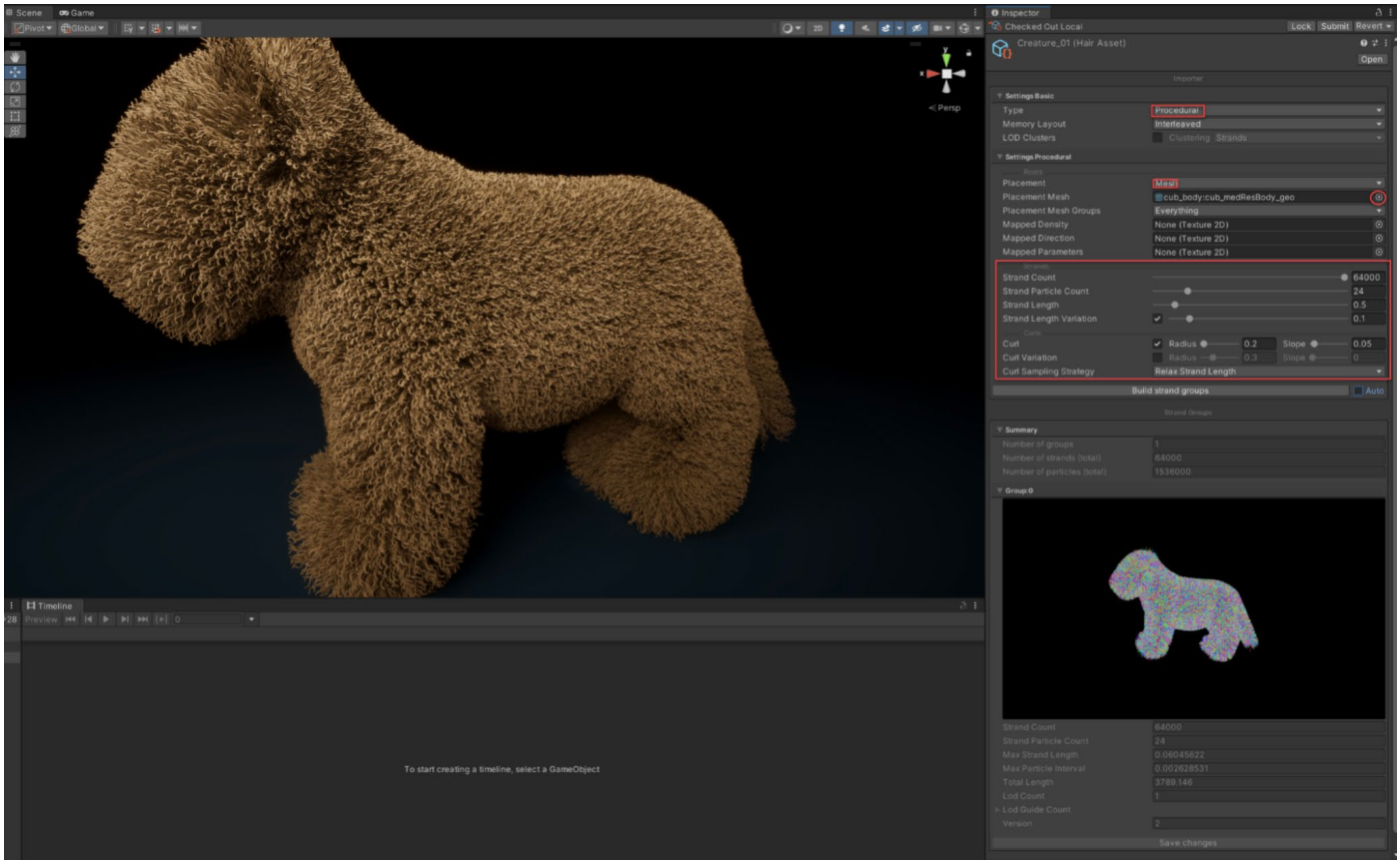
머티리얼 샘플에 포함된 모발 샘플 씬

Scenes 폴더에서 모발 씬을 찾을 수 있습니다. 스트랜드 및 카드 기반 모발 렌더링을 특징으로 인스펙터의 머티리얼 옵션을 포함한 씬입니다.



HDRP Materials Samples 폴더의 사실적인 헤어 셰이더 옵션

이 Unity Learn 튜토리얼을 따라 중력과 힘의 영향을 받는 동적 모발 시스템을 직접 제작해 보셔도 됩니다.

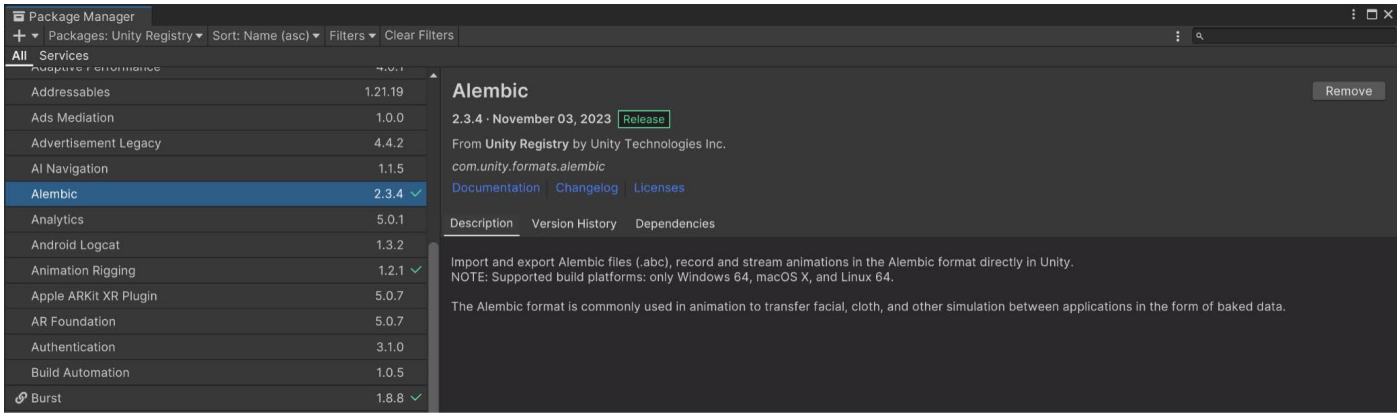


Unity Learn 튜토리얼 '모발 시뮬레이션 시작하기'를 통해 캐릭터와 오브젝트에 자체 모발 시뮬레이션을 추가하는 방법을 알아보세요.

Alembic

Alembic 패키지를 통해 Unity에서 alembic 파일을 사용할 수 있습니다. 이 패키지를 사용하면 사실적인 고급 시뮬레이션을 계산하는 데 필요한 성능 비용을 들이지 않고도 물과 기타 유체, 움직이는 모발과 천이 있는 복잡한 애니메이션을 제작할 수 있는 것입니다. Alembic은 버텍스의 움직임을 플레이 모드에서 재생되는 파일로 베이크합니다.

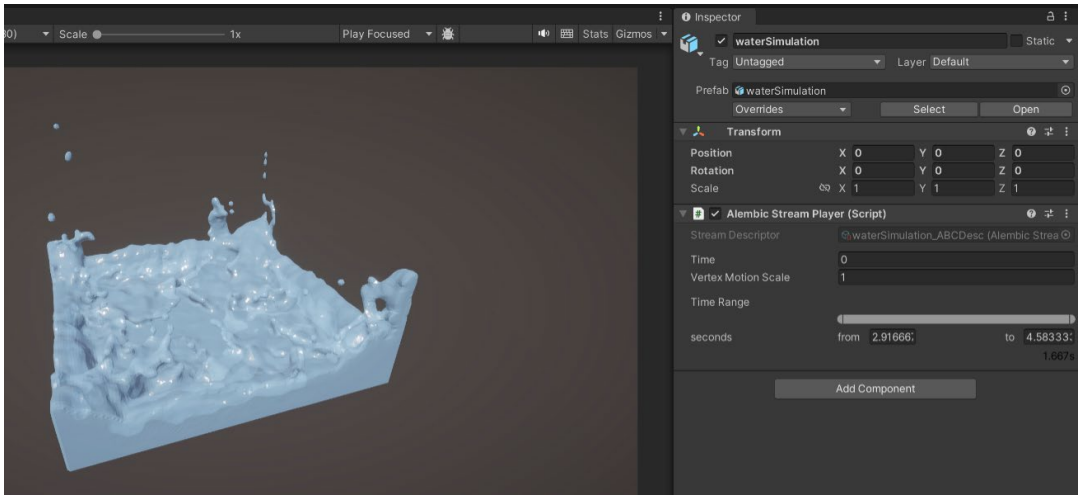
패키지 관리자에서 Alembic 패키지를 설치하여 alembic.abc 파일을 Unity로 임포트합니다.



Unity 패키지 관리자의 Alembic 임포터 및 재생 패키지

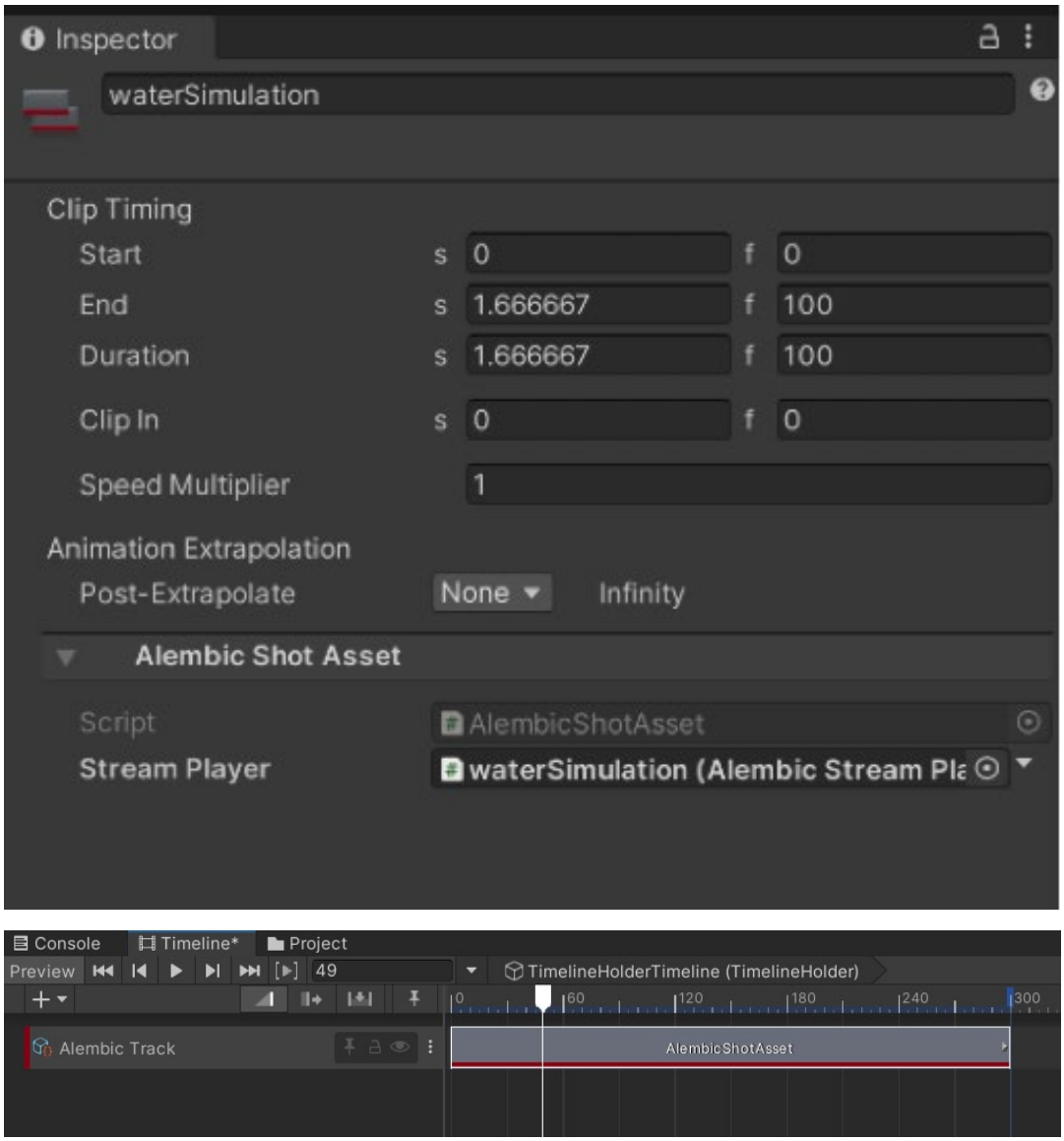
시뮬레이션은 일반적으로 Maya, Blender, Marvelous Designer 등의 다른 소프트웨어에서 제작된 다음 .abc alembic 파일 확장자를 사용하여 익스포트됩니다. Alembic은 프레임당 버텍스 위치를 저장하므로 파일 크기가 매우 클 수 있으며, 시뮬레이션의 복잡성에 따라서는 1GB 이상이 될 수도 있습니다. 따라서 Alembic은 실시간 게임플레이가 아닌 애니메이션 시퀀스에 가장 적합합니다.

프로젝트 창으로 드래그하여 alembic 파일을 임포트한 후 씬에 추가합니다.



Blender에서 생성하고 Unity로 임포트한 Alembic 유체 시뮬레이션

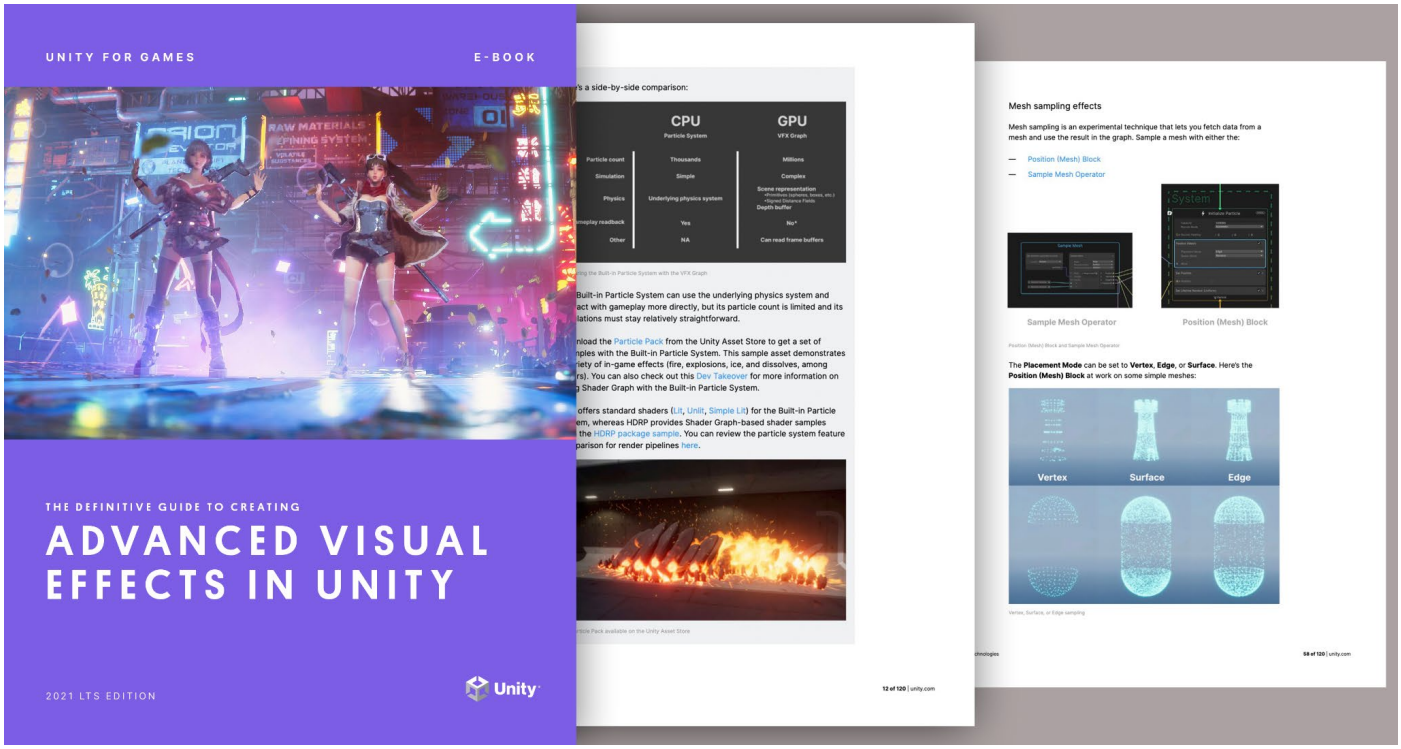
alembic 파일을 재생하려면 Alembic 트랙을 사용하여 타임라인 시퀀스에 오브젝트를 추가합니다.



타임라인에서 Alembic 트랙을 사용하여 alembic 파일 재생

파티클

파티클은 불, 연기, 물, 마법 공격 등과 같은 효과를 구현하는 데 사용됩니다. 파티클은 프레임 속도에 거의 영향을 주지 않으면서 수천 개씩 씬에 렌더링할 수 있어 매우 유용합니다. Unity 에셋 스토어에서 다양한 파티클 효과를 찾아보고 Unity의 고급 시각 효과 집중 탐구 가이드 전자책을 통해 Unity에서 파티클 효과를 만드는 방법을 알아보세요.

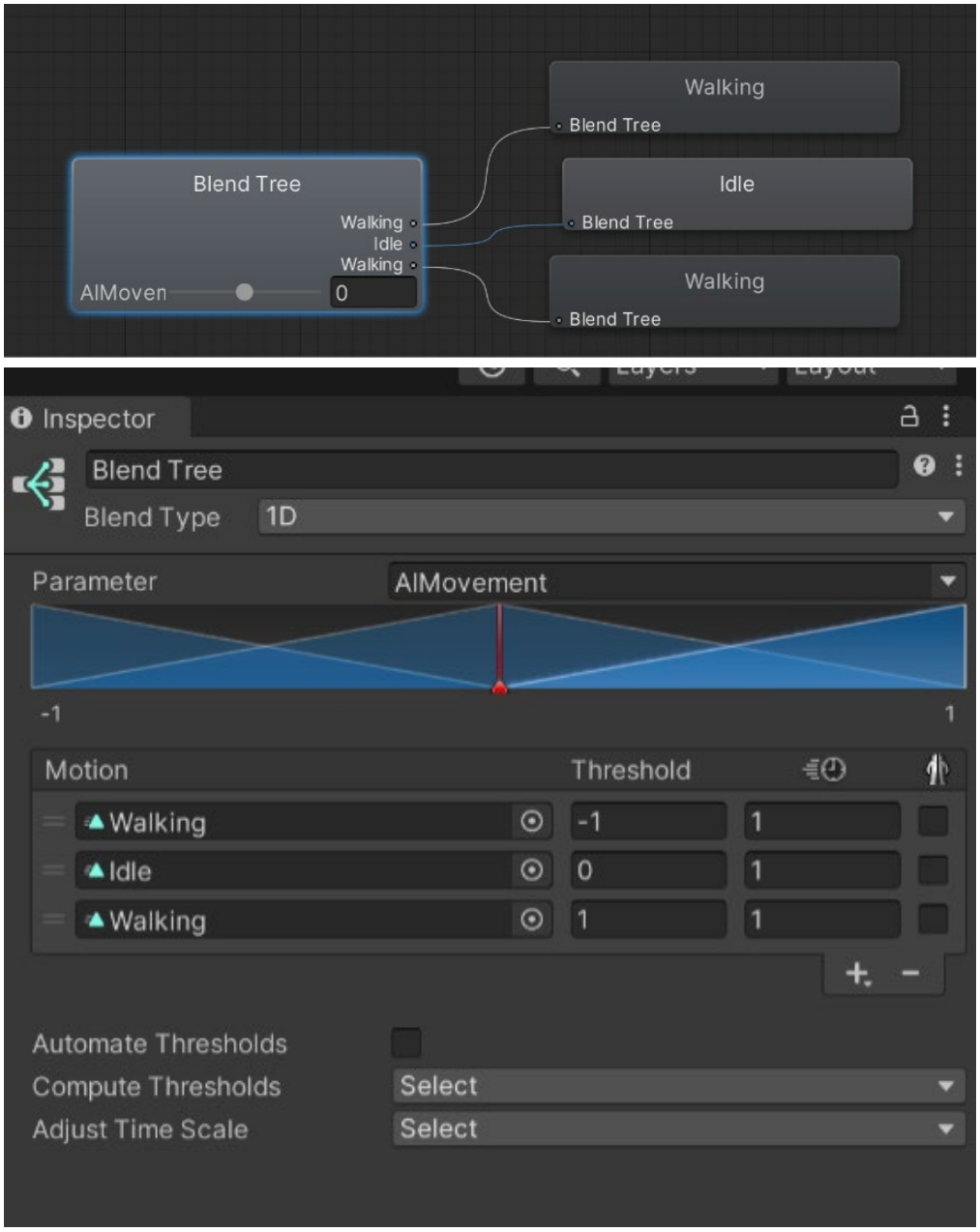


Unity 전자책 다운로드

AI 캐릭터

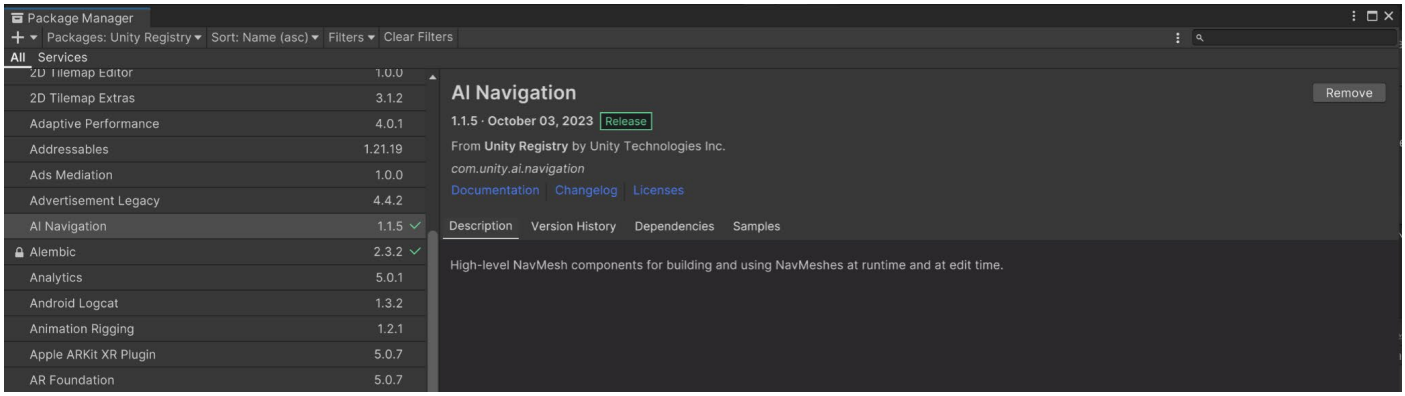
NPC(논플레이어블 캐릭터)를 애니메이션화해야 하거나 씬에서 대규모의 군중을 구현해야 하는 경우, Unity의 AI 내비게이션 시스템을 사용하면 캐릭터를 각각 애니메이션화하는 데 시간을 할애하지 않아도 됩니다. 이 시스템을 사용하면 씬에 등장하는 수많은 캐릭터가 건물이나 오브젝트를 뚫고 지나가거나 서로 부딪히지 않도록 제어할 수 있습니다.

내비게이션 시스템을 시작하려면 -1에서 1 사이의 변수를 사용해 대기 상태와 걷기 상태를 전환하는 방식으로 제어되는 캐릭터의 블렌드 트리를 만듭니다.



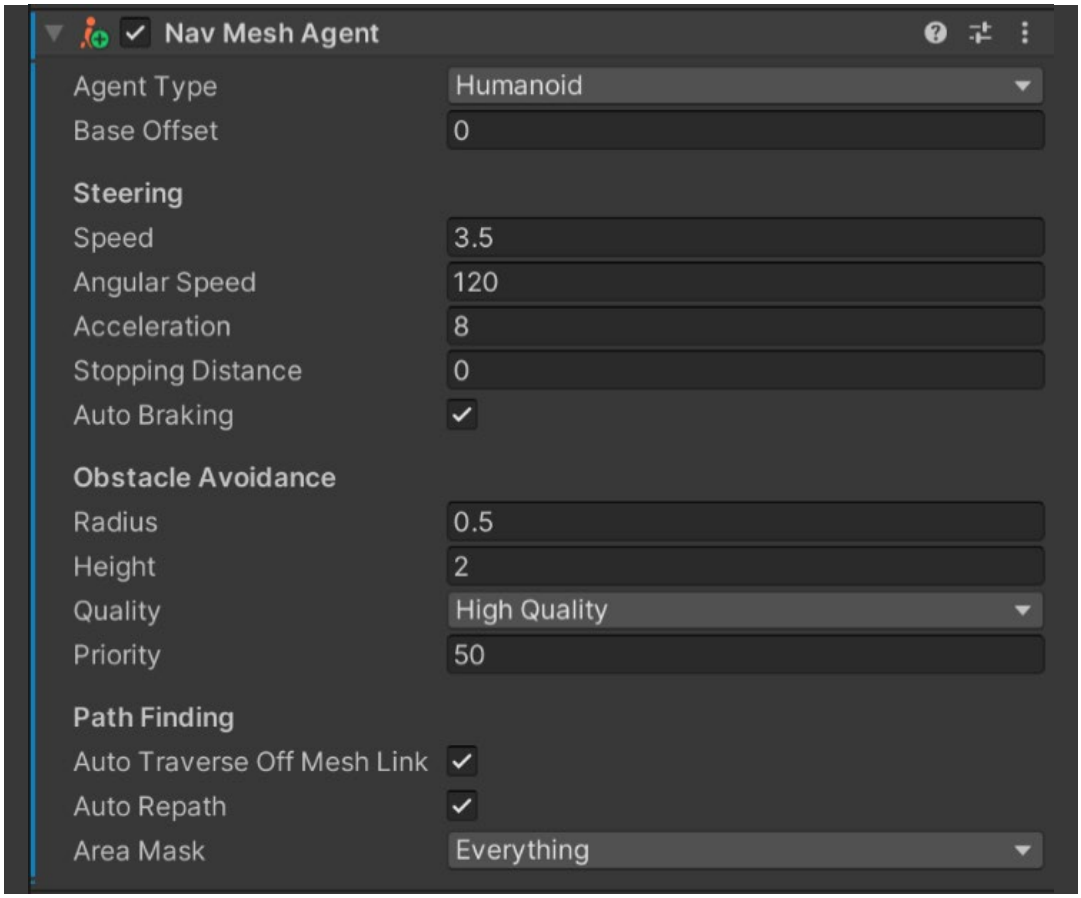
NPC 캐릭터의 블렌드 트리 애니메이터

패키지 관리자에서 AI Navigation 패키지를 설치합니다.



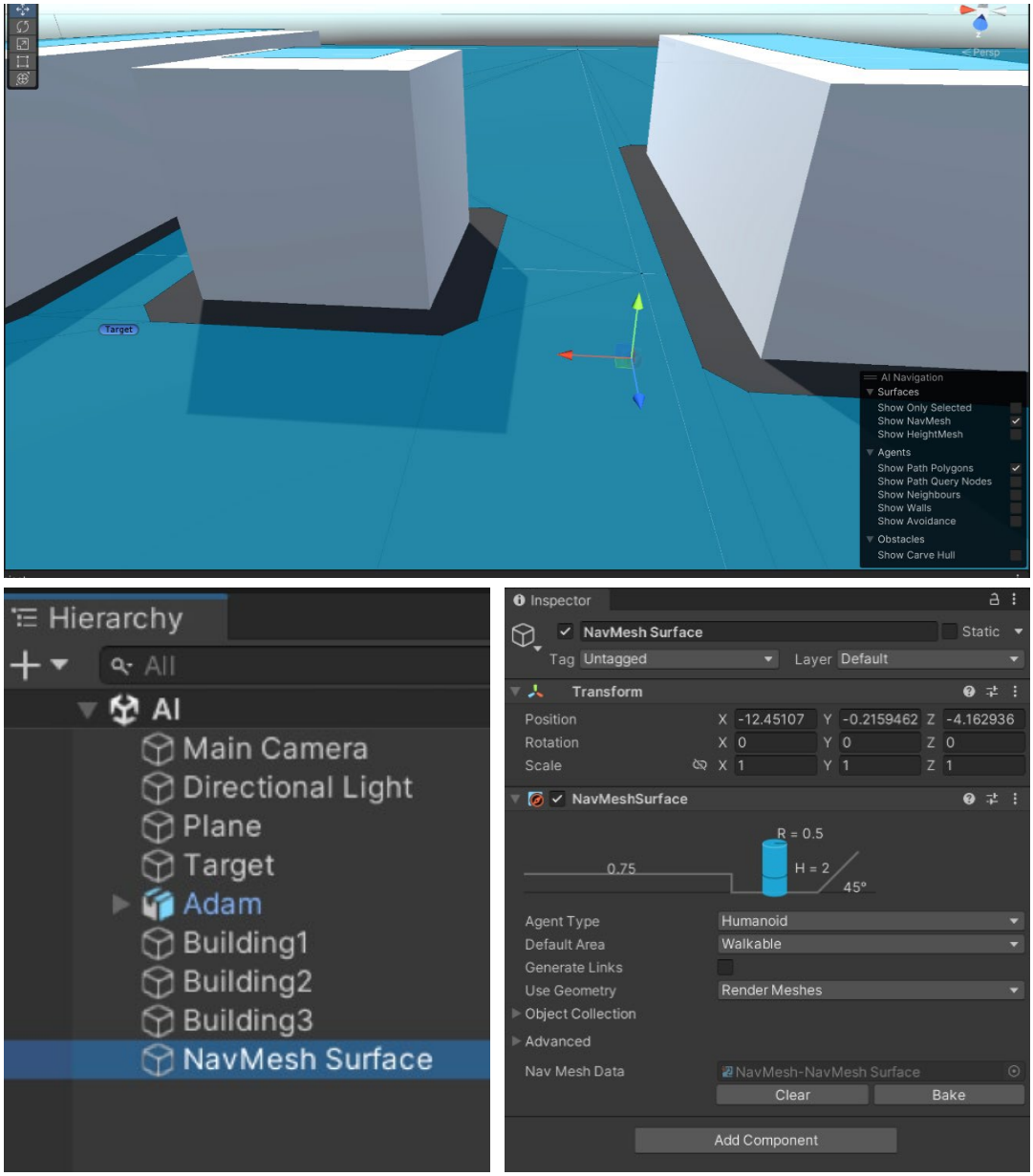
패키지 관리자의 AI Navigation 패키지

이제 NPC에 내비메시(NavMesh) 에이전트를 추가할 수 있습니다. 설정을 변경하여 캐릭터가 걷고 회전하는 속도는 물론, 그 캐릭터에 다른 캐릭터가 얼마나 가까이 다가갈 수 있는지 결정하는 에이전트 반지름도 제어할 수 있습니다.



NPC 제어를 위한 Nav Mesh Agent 설정

캐릭터의 이동을 지시하려면 내비메시 에이전트에 NavMesh Surface 컴포넌트가 있어야 합니다. **GameObject > AI > Navmesh surface**로 이동합니다. 내비메시를 베이크하기 전에 동적 캐릭터를 모두 끕니다. **Show NavMesh** 옵션을 선택하면 내비메시가 파란색 표면으로 표시됩니다. NPC가 걸어갈 타겟 오브젝트를 추가합니다.



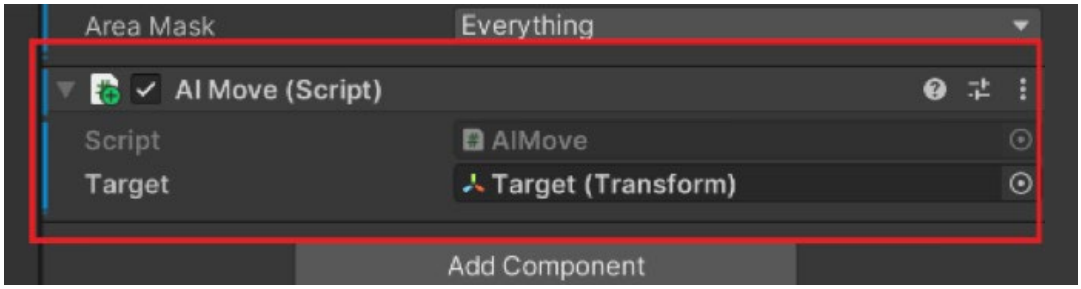
내비메시가 베이크되어 씬에 표시되어 있으며 NPC는 파란색 영역에서만 걸을 수 있음

내비메시 에이전트를 제어하려면 스크립트가 필요합니다. 이 스크립트는 에이전트가 움직일 때 걷기를 재생할지, 멈췄을 때 대기 상태로 설정할지 에이전트의 속도를 활용하여 애니메이터에 알려 줍니다.

```
AIMove.cs
Assembly-CSharp                                    AIMove
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.AI;

Unity Script | 0 references
public class AIMove : MonoBehaviour
{
    private NavMeshAgent agent;
    private Animator animator;
    private float characterMovement;
    private float moveSpeed;
    public Transform target;

    // Start is called before the first frame update
    Unity Message | 0 references
    void Start()
    {
        agent = GetComponent<NavMeshAgent>();
        animator = GetComponent<Animator>();
    }

    // Update is called once per frame
    Unity Message | 0 references
    void Update()
    {
        agent.destination = target.position;
        characterMovement = agent.velocity.x + agent.velocity.z;
        animator.SetFloat("AIMovement", characterMovement);
    }
}
```

NPC의 내비메시 에이전트를 제어하는 스크립트

타겟을 이 스크립트의 슬롯으로 드래그합니다. 타겟과 캐릭터는 원하는 수만큼 사용할 수 있습니다. 가급적이면 캐릭터당 하나의 타겟을 지정하는 식으로 설정을 단순하게 유지하세요.



NPC는 항상 타겟을 향해 걸어가다가 타겟에 도달하면 멈춥니다.

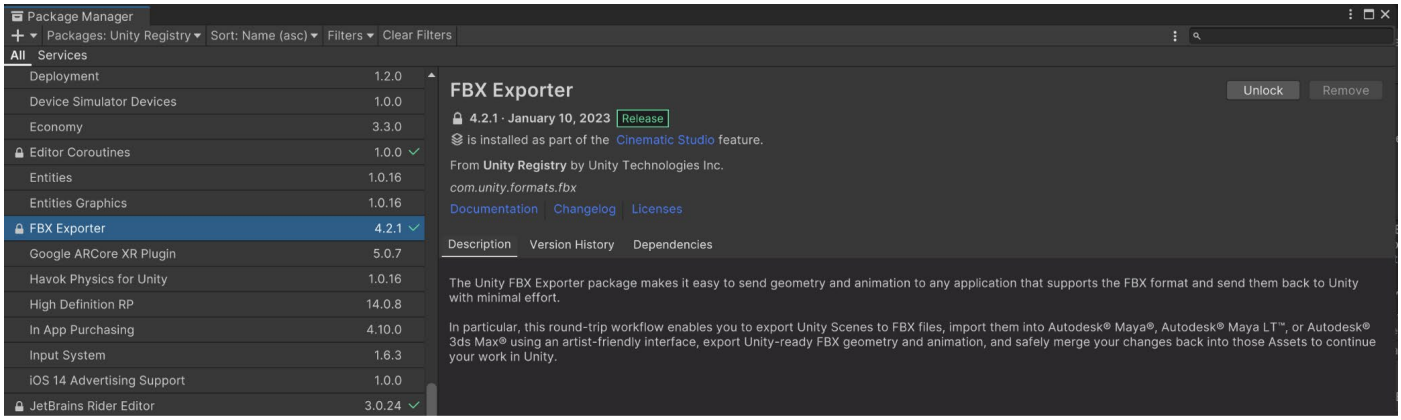
이제 플레이 모드를 실행하면 캐릭터는 타겟을 향해 걸어갑니다. 움직이는 타겟을 애니메이션화하여 NPC가 따라가게 구현할 수도 있습니다. AI 내비게이션을 사용하여 전체 군중 시뮬레이션을 얼마나 쉽게 구현할 수 있는지 알아보세요.

애니메이션 익스포트

팀 단위로 작업하거나 DCC 툴에서 애니메이션을 다듬고 싶은 경우 애니메이션화된 캐릭터를 Unity에서 다른 소프트웨어로 익스포트하여 조정해야 하는 경우가 많습니다.

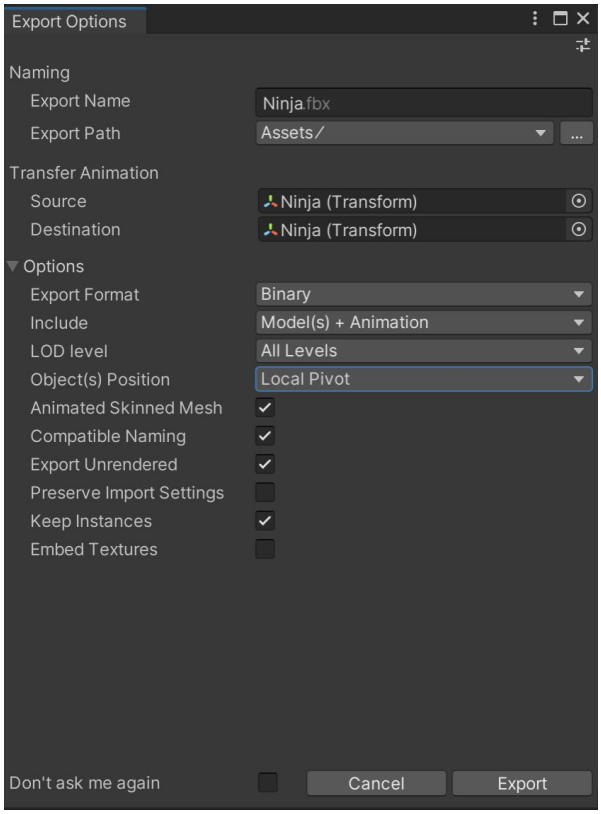
FBX 모델 익스포트

패키지 관리자의 Registry에서 FBX Exporter가 설치되어 있어야 합니다.



FBX 파일을 익스포트하려면 FBX Exporter를 설치해야 합니다.

캐릭터는 Animator 창에서 애니메이션이 설정된 상태여야 합니다. 캐릭터를 오른쪽 클릭하고 **Export to FBX**를 선택합니다.



FBX 익스포트 옵션

익스포트 상자가 열리고 대상 폴더를 선택할 수 있습니다. 기본 대상 폴더는 Project Assets 폴더입니다.

FBX 파일을 열 소프트웨어에 따라 ASCII 또는 바이너리 포맷으로 파일을 익스포트할 수 있습니다. Blender의 경우 Binary를 선택합니다.

이때 파일은 3D 메시거나 애니메이션 파일이거나 둘 다일 수 있습니다.

LOD 수준이 있는 경우 모두 익스포트하거나 최고 또는 최저 디테일 수준(LOD) 중 하나를 선택합니다.

Animated Skinned Mesh를 선택하여 스킨드 지오메트리가 소프트웨어에 표시되도록 합니다. 텍스처도 다른 소프트웨어에 표시하려면 **Embed Textures** 옵션을 선택합니다.

익스포트 후에는 다른 3D 소프트웨어에서 파일을 열어 3D 메시 또는 애니메이션 파일을 조정하면 됩니다.

Blender에서 익스포트되었으며 애니메이션이 포함된 FBX 파일

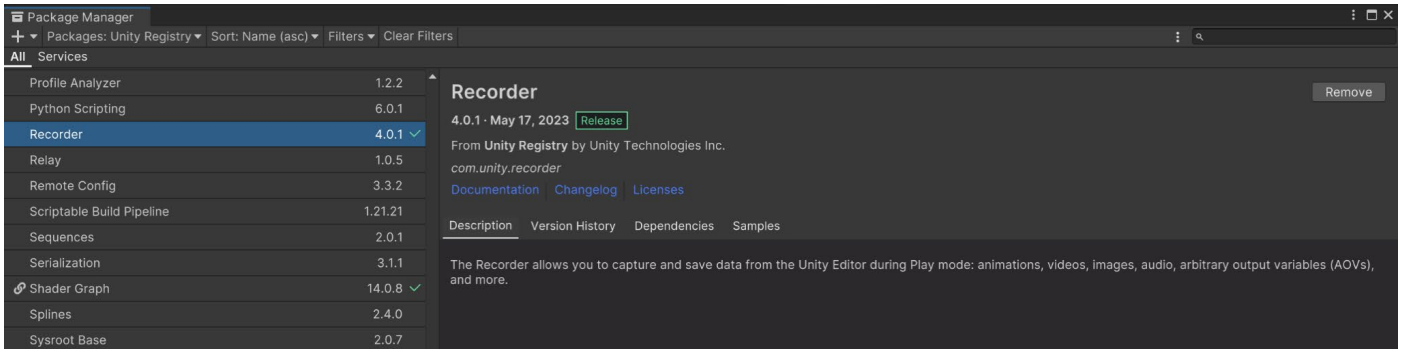
임베디드 애니메이션은 Blender의 비선형 애니메이션 창에서 사용할 수 있습니다. 파일을 조정한 후에는 FBX 파일로 익스포트하여 Unity에서 열 수 있습니다.

Unity Recorder

Unity로 애니메이션 영화를 제작하거나 고품질 게임 영상이 필요한 경우 최종 애니메이션을 동영상 파일로 익스포트합니다.

에디터 또는 빌드에서 게임을 플레이하면서 원본 영상을 캡처하는 동시에 동영상 캡처 소프트웨어를 실행하면 컴퓨터 리소스에 부하가 걸려, 게임이 마케팅이나 동영상 콘텐츠 용도에 부합할 만큼의 품질로 실행되지 않을 수 있습니다. 프레임 건너뛰기로 인한 불안정 현상 없이 안정적인 프레임 속도를 확보하고 최상의 레코딩 품질을 얻기 위해 Unity Recorder 를 사용하면 에디터 내에서 실시간으로 동영상 또는 이미지 시퀀스를 제작할 수 있습니다.

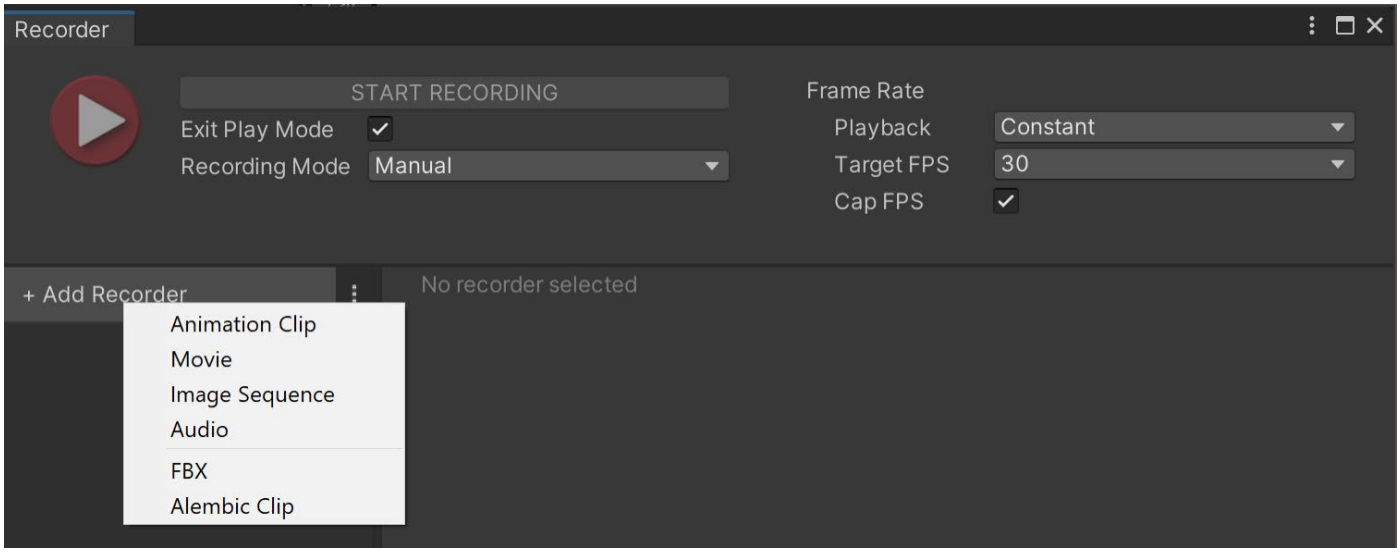
패키지 관리자에서 Recorder를 설치합니다.



동영상 및 이미지 시퀀스를 익스포트하기 위한 Recorder 패키지

씬을 설정하고 녹화할 준비가 되면 **Window > General > Recorder > Recorder 창**으로 이동합니다. 설정 목록에서 프레임 속도를 설정하거나 직접 정의할 수 있습니다. 더욱 안정적인 동영상 파일을 생성하기 위해 비트레이트를 일정하게 유지합니다.

새로운 녹화본을 생성하려면 **Add Recorder** 탭을 클릭하고 제공되는 옵션 중에서 선택합니다. 동영상 및 이미지 시퀀스를 생성하거나 오디오를 익스포트할 수 있습니다.



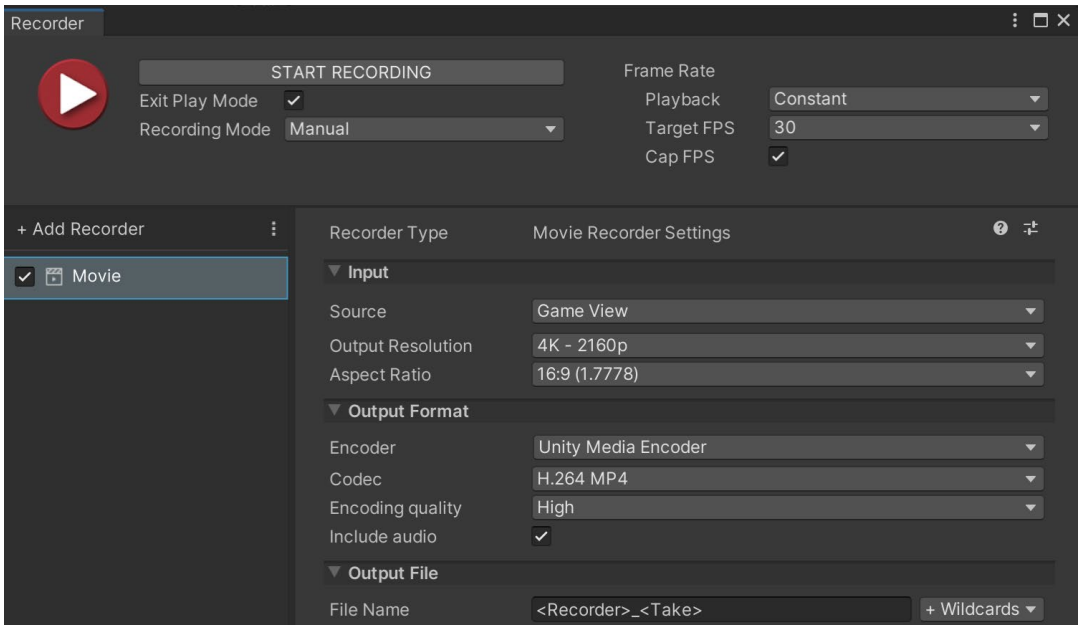
새 레코더 만들기

Movie 옵션을 선택하고 익스포트 요구 사항에 맞게 설정을 커스터마이즈합니다. 이 동영상은 씬 뷰에서 녹화되므로 씬 뷰에 표시되는 모든 내용이 녹화됩니다. 특정 카메라에서만 동영상을 녹화하거나 VR용 360° 뷰를 녹화할 수도 있습니다.

설정 목록에서 출력 해상도를 설정하거나 직접 정의할 수도 있으며, 그러면 녹화 시 씬 뷰가 해당 해상도를 따릅니다. 4:3처럼 다른 비율로 씬이 설정되어 있는 경우 16:9 출력을 선택하면 녹화 시에 씬 뷰가 16:9로 변경되므로 동영상이 에디터의 씬과는 다르게 보일 수 있습니다.

Output Format에서는 사용할 인코더를 선택할 수 있습니다. Unity Media Encoder는 H.264 MP4 및 WebM 코덱을 지원합니다. Apple ProRes 인코더는 Apple ProRes 4444 및 422 코덱을 지원합니다. Gif 인코더는 애니메이션 gif를 익스포트하는 데 사용됩니다.

동영상 익스포트 시 오디오를 포함하거나 제외하도록 설정할 수 있으며, 파일은 프로젝트의 recordings라는 폴더에 저장됩니다.



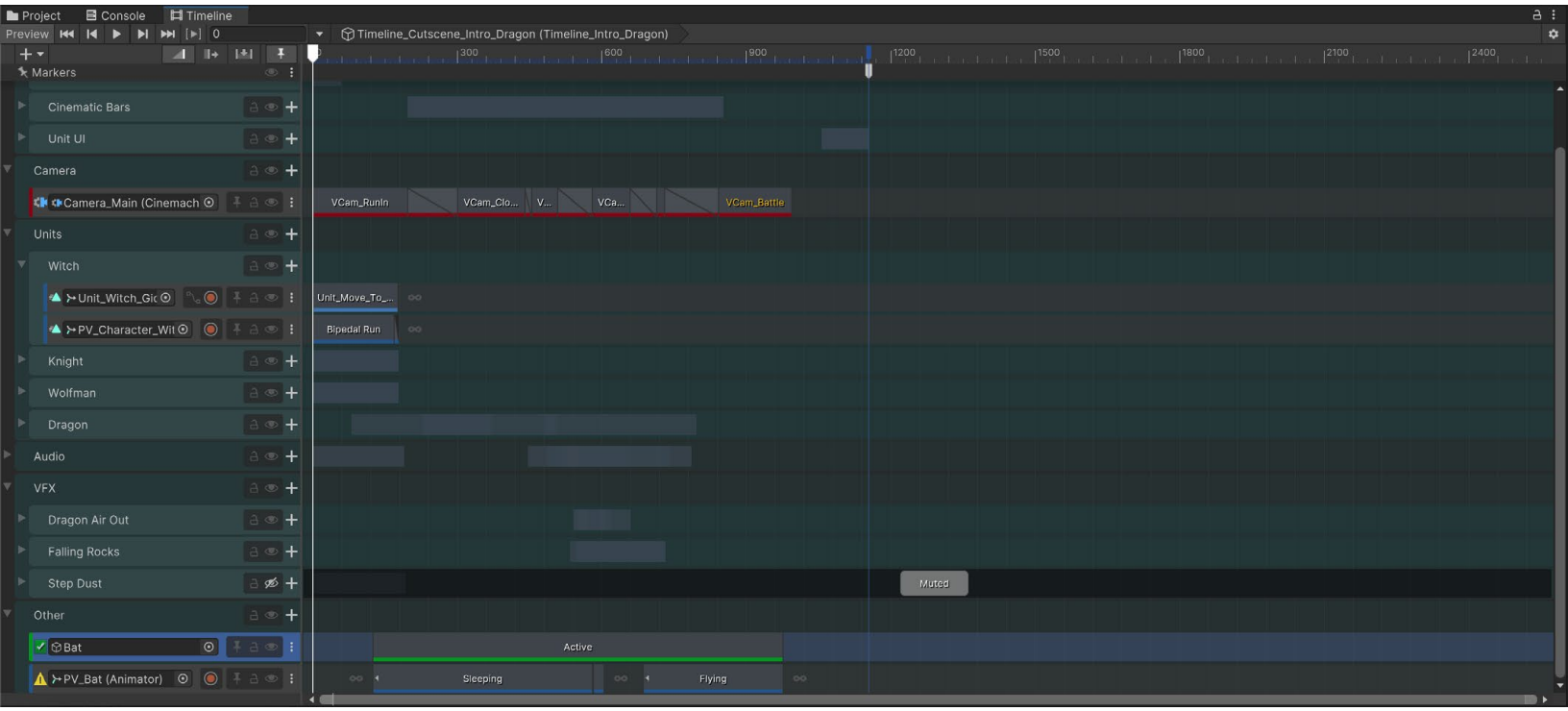
동영상 파일 익스포트를 위한 Movie 설정

프로젝트를 JPG, PNG, EXR 파일로 된 이미지 시퀀스로 익스포트할 수도 있습니다. 이러한 파일은 컬러 그레이딩과 동영상 효과를 추가한 후 동영상 편집 소프트웨어에서 최종 인코딩을 수행하려는 경우 유용할 수 있습니다.

Start Recording을 눌러 녹화 모드로 전환하면 동영상이 녹화되면서 시퀀스 재생이 시작됩니다. 이때 재생 화면이 타겟 프레임 속도로 제한되고, 타겟 프레임 속도보다 느리게 재생될 수도 있습니다. 이는 인코딩 프로세스로 인한 현상으로, 동영상 자체는 타겟 프레임 속도로 부드럽게 재생됩니다.

Recording Mode가 Manual로 설정된 경우 적당한 시간에 정지를 눌러야 합니다. Recording Mode 설정에서 녹화의 시작 프레임과 종료 프레임을 입력할 수 있습니다.

타임라인을 사용하여 레코더 트랙을 추가할 수도 있습니다. 트랙을 오른쪽 클릭하여 RecorderClip을 추가합니다. 이 트랙은 활성 블록처럼 작동하며 해당 블록의 지속 시간 동안 녹화가 진행됩니다.



타임라인을 사용하여 레코더 시작 지점과 정지 지점 설정

2D 애니메이션

Unity의 2D 프로젝트도 비슷한 툴과 워크플로를 공유합니다. 2D 프로젝트에서 FBX 3D 모델에 해당하는 것은 2D 스프라이트입니다. 2D 스프라이트 에디터, 2D 타일맵 시스템, 2D 스프라이트 셰이프뿐만 아니라 2D 물리 및 조명 툴셋도 있습니다. Animator 컨트롤러와 타임라인 툴도 2D 프로젝트에 사용됩니다.

기존의 2D 애니메이션은 2D 게임 개발 과정에서 가장 까다롭고 시간이 많이 걸리는 부분일 수 있습니다. 2D 골격 애니메이션을 사용하면 시간을 절약하고, 다양하게 활용할 수 있으면서도 부드러운 애니메이션을 만들 수 있으며, 더 적은 에셋으로 더 많은 애니메이션을 구현할 수 있습니다.

PSD 워크플로

2D Animation 패키지를 사용하면 캐릭터 아트워크를 Photoshop에서 Unity로 바로 임포트할 수 있습니다. 2D PSD Importer 패키지는 이러한 목적으로 레이어가 적용된 PSD와 PSB 파일을 처리합니다. 캐릭터의 모든 레이어를 스프라이트로 임포트하여 앱에서 그려 둔 대로 배치할 수 있습니다.

임포트한 PSD 또는 PSB 파일을 선택하면 스프라이트 임포트 설정과 비슷한 옵션이 2D 애니메이션 및 리깅과 관련된 추가 설정과 함께 표시됩니다. 선택해야 할 주요 옵션은 다음과 같습니다.

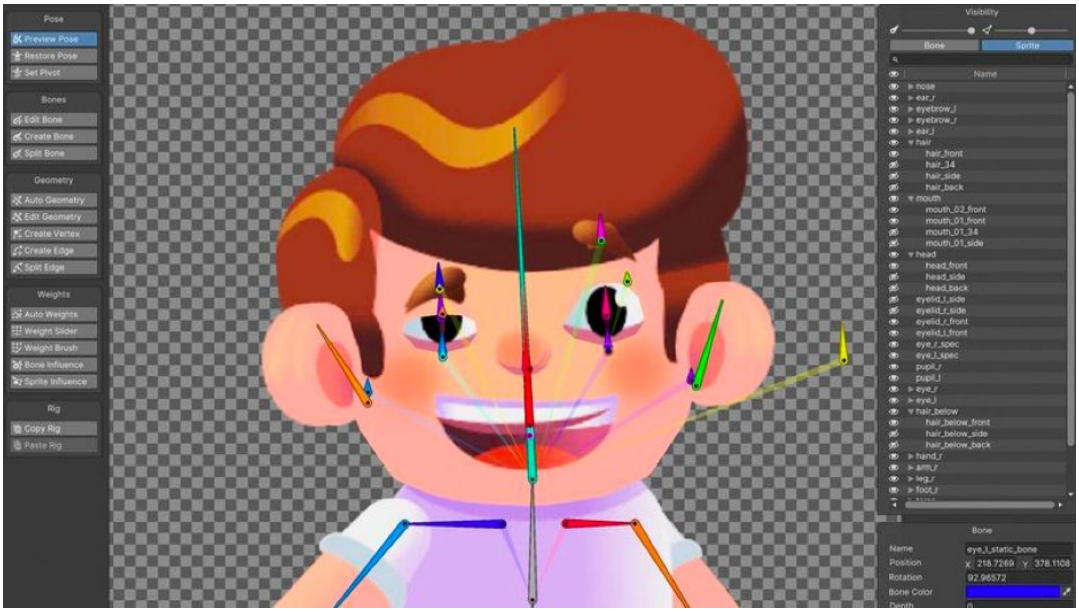
— **Import Hidden Layers:** 이 옵션은 숨겨진 레이어를 포함하여 PSD 또는 PSB 파일의 모든 레이어를 임포트합니다.

— **Individual sprites (Mosaic):** 이 설정은 Texture Type이 Multiple로 설정된 경우에만 사용할 수 있습니다. 임포트한 레이어에서 스프라이트를 만들고 텍스처 아틀라스에 배치하는 설정입니다. 캐릭터를 리깅하려면 이 옵션을 활성화한 상태로 두세요.

— **Use as Rig:** 이 옵션은 PSD 또는 PSB 파일과 동일한 레이어 계층 구조 및 위치를 유지하며 캐릭터 프리팹을 생성합니다.

— **Use Layer Group:** PSD 또는 PSB 파일에서 레이어 그룹을 추가합니다. 이 옵션을 활성화하여 스프라이트가 교체된 부위 등 캐릭터의 부위를 그룹화할 수 있습니다.

Layer Management 섹션에서 레이어의 계층 구조를 확인하고 임포트할 레이어를 수동으로 추가하거나 제거할 수 있습니다.



해피 하비스트 샘플 프로젝트의 주인공

스프라이트 에디터에서의 리깅

스프라이트 에디터 내부의 Skinning Editor에서 캐릭터의 골격을 만들 수 있습니다. 먼저 Create Bone 버튼을 클릭한 후, 메인 창 영역을 클릭합니다. 첫 번째 클릭으로 뼈대의 중심점이 생성되며 두 번째 클릭으로 뼈대의 끝 위치가 표시됩니다. 이 툴은 뼈대를 서로 연결하고 중첩시킵니다.

Edit Bone 버튼을 사용하여 뼈대를 조정합니다. Split Bone 버튼을 사용하면 뼈대를 둘로 나눌 수 있습니다. 팔다리를 만들 때 유용한 옵션입니다. 다리 뼈대를 하나 만들고 무릎이 될 곳을 클릭하면 뼈대가 허벅지 부위와 종아리 부위로 나뉩니다.

계층 구조가 올바른지 확인하려면 Preview Pose 버튼을 선택하고 몇 가지 포즈를 테스트합니다. 뼈대의 위치를 재설정하려면 툴바에서 Reset Pose 버튼을 누릅니다.

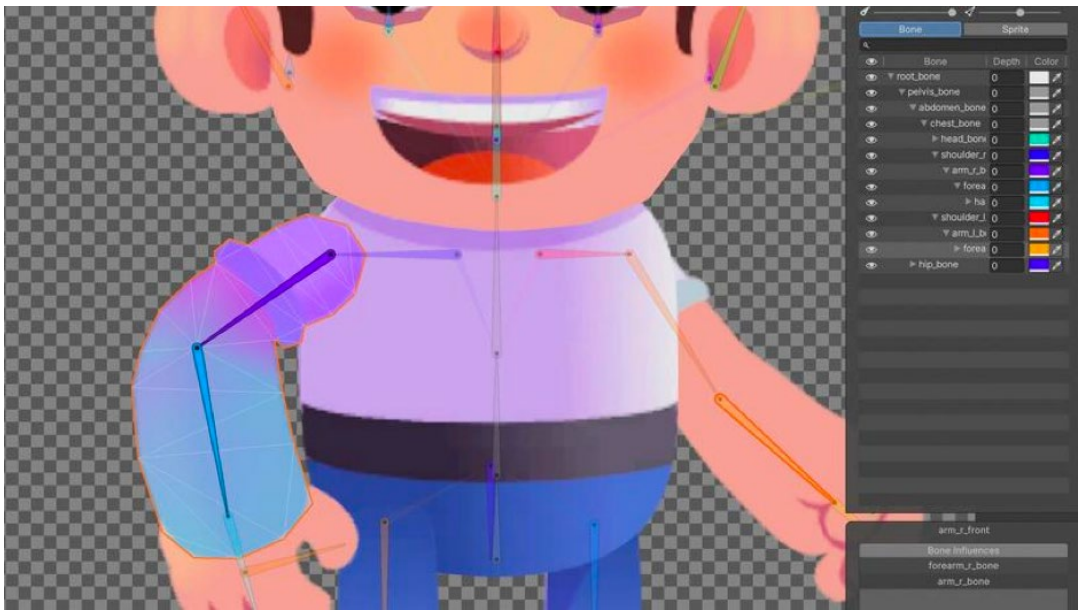
스프라이트를 뼈대, 지오메트리, 가중치에 연결

뼈대에 스프라이트를 할당하려면 지오메트리를 생성해야 합니다. 먼저 Auto Geometry 버튼을 누릅니다. 작은 팝업 창이 열리면 지오메트리 생성 방식을 정의할 수 있습니다.

모든 슬라이더를 0으로 설정하여 지오메트리를 최대한 간단히 유지하는 것이 좋습니다. Weights 옵션을 활성화하여 뼈대를 스프라이트에 자동으로 바인딩합니다. Generate For All Visible 버튼을 클릭하면 모든 스프라이트에 대한 뼈대의 가중치가 생성 및 설정됩니다. 이를 개별 스프라이트에 적용하려면 스프라이트를 더블 클릭합니다. 특정 스프라이트의 지오메트리를 조정하는 데 유용한 방법입니다.

다양한 버텍스와 지오메트리 굴절 방식까지 전부 제어하려면 Geometry 섹션에서 툴로 메시를 수동으로 편집해야 합니다.

가급적 적은 수의 버텍스를 사용하세요. 뼈대의 회전에 따라 모든 버텍스 위치를 계산해야 하므로 버텍스 수가 적으면 리소스를 절약할 수 있습니다. 점의 개수가 적을수록 가중치 설정이 더 수월하므로 버텍스 수가 적으면 메시의 굴절도 더 개선됩니다. 또한 최적의 게임 버텍스 수는 타겟 플랫폼마다 다르니 지오메트리는 항상 최적화하는 것이 좋습니다.



애니메이션을 최적화하려면 지오메트리를 명확하게 설정하는 것이 중요하며, 특히 관절처럼 변형이 크게 일어나는 영역에서는 더욱 그렇습니다.

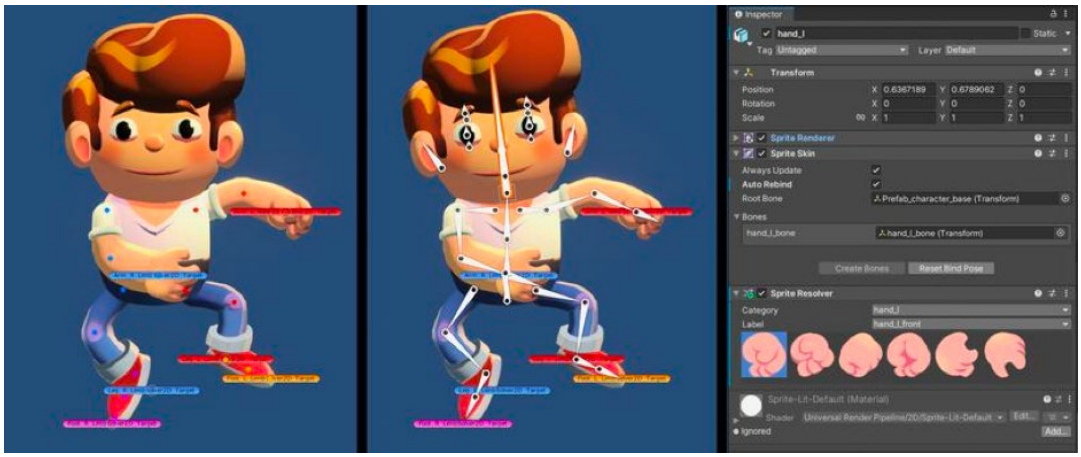
가중치

스프라이트의 지오메트리를 깔끔하게 정리했다면 이제 가중치를 설정할 차례입니다. 가중치는 뼈대가 각 버텍스에 미치는 영향을 0에서 1로 정의합니다. 0은 뼈대가 버텍스에 미치는 영향이 없음을, 1은 버텍스가 뼈대에 붙은 듯이 움직이는 것을 의미합니다. 적절한 가중치가 설정되면 메시의 굴절을 사실적으로 표현할 수 있습니다. 가중치를 잘못 설정하면 게임의 몰입감을 깨뜨리고 스프라이트를 왜곡시킬 수 있습니다.

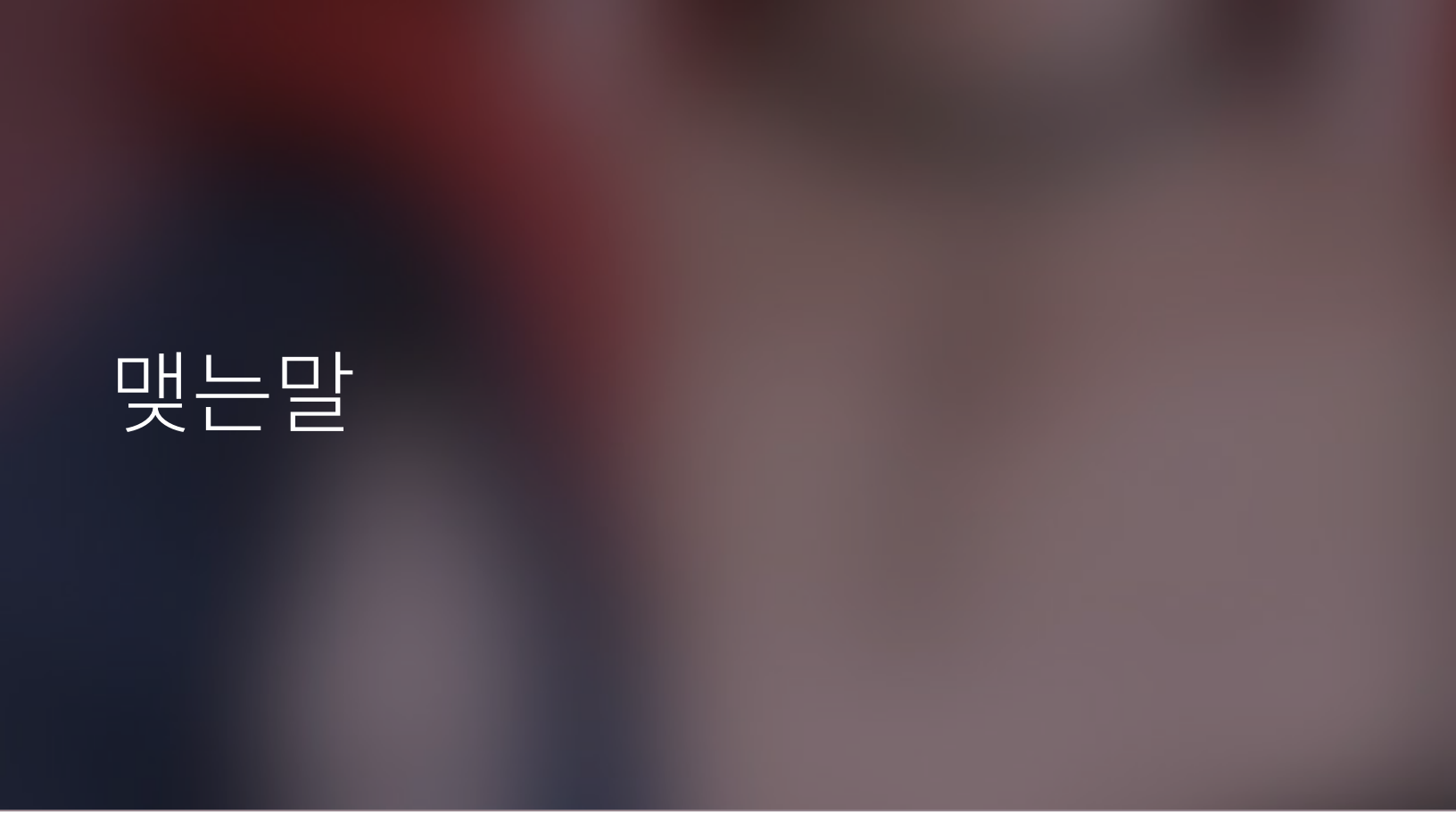
가중치 설정을 시작하려면 특정 스프라이트에 영향을 미칠 뼈대를 정의해야 합니다. Bone Influence 버튼을 클릭하고 스프라이트를 선택합니다. 작은 팝업 창이 표시되면 선택한 스프라이트에 영향을 미치는 뼈대를 설정할 수 있습니다. 스프라이트에 영향을 주기 위해 필요한 뼈대만 설정하고 나머지는 제거합니다.

스프라이트 리졸버 및 2D IK

해피 하비스트(Happy Harvest) Unity 샘플 프로젝트에서는 여러 스프라이트가 동일한 뼈대의 영향을 받는 것을 확인할 수 있습니다. 예를 들어 손 스프라이트는 애니메이션 프로세스 또는 게임플레이 중에 측면, 앞면, 뒷면 등의 다양한 형태로 교체될 수 있습니다. 또한 게임 월드에 반응하는 애니메이션을 제작하려는 경우 2D 역운동학(IK)을 사용하면 회전을 계산하고 연결된 뼈대를 타겟 위치로 이동할 수 있습니다.



2D IK 솔버를 사용해 해피 하비스트의 주인공을 애니메이션화합니다. 왼쪽 이미지는 뼈대의 위치와 회전을 수정하는 모습을 보여 줍니다. 오른쪽 이미지는 스프라이트 리졸버로 손의 스프라이트를 변경하는 모습을 보여 줍니다.



맺는말

Unity로 애니메이션을 제작하고자 하는 아티스트라면 전체 Unity 기술 자료, Unity Learn, Unity 블로그, 애니메이션 포럼을 확인하여 필요한 정보를 참고하세요.

Unity 베스트 프랙티스 허브 또는 Unity 매뉴얼의 베스트 프랙티스 섹션에서 Unity에 대한 심층적인 내용이 담긴 개발자 및 아티스트용 전자책을 모두 확인할 수 있습니다.

Unity 제품 보드 에서는 향후 출시될 기능과 현재 개발 중인 애니메이션 기능의 개요를 볼 수 있습니다. 원하는 기능을 직접 요청할 수도 있습니다.

게임 개발에서 좋은 성과가 있기를 기원합니다.

unity.com/kr