언리얼 에디터 인터페이스

1. 메뉴 바
메뉴를 사용하여 에디터 특정 명령과 함수 기능에 액세스할 수 있습니다.

2. 메인 툴바
언리얼 엔진의 가장 일반적인 툴과 에디터 일부를 위한 단축 키를 포함하고 있으며, 이 단축키를 통해 플레이 모드를 입력하고 프로젝트를 기타 플랫폼에 디플로이할 수 있습니다.
1) 메인 툴바 구성 - 저장 버튼, 모드 선택, 콘텐츠 단축키, 플레이 모드 컨트롤, 플랫폼 메뉴, 세팅

3 레벨 뷰포트
카메라, 액터, 스태틱 메시 등과 같은 레벨 콘텐츠를 표시합니다.
레벨 뷰포트 콘텐츠 표시 방법
1) 원근 - 다른 각도에서 뷰포트의 콘텐츠를 보기 위해 탐색할 수 있는 3D 뷰
2) 직교 - 주요 축(X, Y 또는 Z) 중 하나를 내려다보는 2D 뷰

4. 콘텐츠 드로어 버튼
프로젝트의 모든 에셋에 액세스할 수 있는 콘텐츠 드로어 를 엽니다.

5. 하단 툴바
명령 콘솔, 출력 로그, 파생 데이터 기능의 단축키가 포함되어 있습니다. 소스 컨트롤 상태도 표시됩니다.

6. 아웃라이너
레벨의 모든 콘텐츠에 대한 계층 트리 뷰를 표시
각각 고유한 열 레이아웃과 필터 환경설정이 있는 최대 4개의 서로 다른 아웃라이너를 사용할 수 있다.

7. 디테일 패널
액터 선택 시 나타내며 트랜스폼(레벨의 위치), 스태틱 메시, 머티리얼, 피직스 세팅과 같은 해당 액터를 위한 다양한 프로퍼티를 표시합니다. 이 패널은 레벨 뷰포트에서 무엇을 선택하는지에 따라 다른 세팅을 표시합니다.

언리얼 엔진 용어

1. 프로젝트
언리얼 엔진 5 프로젝트는 게임의 모든 콘텐츠를 담고 있습니다. 프로젝트에는 Blueprints , Materials 등 하드디스크 안의 많은 폴더가 포함되며 프로젝트 안의 폴더는 원하는 대로 이름을 변경하고 정리할 수 있습니다.

2. 블루프린트
블루프린트 비주얼 스크립팅 시스템은 노드 기반의 인터페이스를 사용하여 언리얼 에디터 내의 게임플레이 엘리먼트를 생성하는 완전한 게임플레이 스크립팅 시스템입니다. 블루프린트 비주얼 스크립팅도 엔진에서 오브젝트 지향 클래스 또는 오브젝트를 정의하기 위해 사용됩니다.

3. 오브젝트
오브젝트 는 언리얼 엔진에서 가장 기본적인 클래스입니다. 오브젝트는 구성단위 역할을 하며, 에셋에 필수적인 기능을 많이 포함하고 있습니다. 언리얼 엔진에서는 거의 모든 것이 오브젝트로부터 상속받거나 함수 기능을 사용합니다

4. 클래스
클래스는 언리얼 엔진에서 특정 액터나 오브젝트의 행동과 프로퍼티를 정의합니다. 클래스는 계층적입니다. 즉, 정보를 부모 클래스(해당 클래스가 파생된 클래스)로부터 상속받고 자손 클래스에게 다시 상속합니다. 클래스는 C++ 코드나 블루프린트로 생성될 수 있습니다.

5. 액터
액터는 카메라, 스태틱 메시, 플레이어 스타트 위치 등과 같이 레벨에 배치할 수 있는 모든 오브젝트를 가리킵니다. 액터는 이동, 회전, 스케일링 등의 3D 트랜스폼을 지원합니다. 또한 C++ 또는 블루프린트 게임플레이 코드를 통해 생성(스폰) 및 소멸될 수 있습니다.

C++에서 AActor는 모든 액터의 베이스 클래스입니다.

6. 형변환
형변환은 특정 클래스의 액터를 다른 클래스인 것처럼 간주하려 시도하는 작업입니다. 형변환한 액터의 클래스 전용 기능에 액세스할 수 있습니다.

7. 컴포넌트
컴포넌트는 액터에 추가할 수 있는 기능 조각입니다.
액터에 컴포넌트를 추가하면, 액터는 해당 컴포넌트가 제공하는 기능을 사용할 수 있습니다.

8. 폰
폰 은 액터의 서브클래스이며, 인게임 아바타 또는 페르소나(게임 안의 캐릭터 등)의 역할을 합니다. 폰은 플레이어 또는 게임 AI 로 제어되는 NPC(논플레이어 캐릭터) 등이 있습니다.

9. 캐릭터
캐릭터 는 플레이어 캐릭터로 사용하기 위한 폰 액터의 서브클래스입니다. 캐릭터 클래스에는 콜리전 설정, 바이페드 움직임을 위한 입력 바인딩, 플레이어가 제어하는 움직임을 위한 추가 코드 등이 포함됩니다.

10. 플레이어 컨트롤러(Player Controller)
플레이어 컨트롤러 는 플레이어의 입력을 게임 안의 상호작용으로 변환합니다. 모든 게임에는 최소 하나의 플레이어 컨트롤러가 있습니다. 플레이어 컨트롤러는 게임 안에서 플레이어를 나타내는 폰이나 캐릭터에 빙의할 때가 많습니다.

11. AI 컨트롤러
플레이어 컨트롤러가 게임 안에 플레이어를 나타내는 폰을 소유하듯이, AI 컨트롤러 도 게임 안에 NPC를 나타내는 폰을 소유합니다. 기본적으로 베이스 AI 컨트롤러에 빙의됩니다.

12. 플레이어 스테이트
플레이어 스테이트 게임 참여자의 스테이트를 말합니다. 인간 플레이어 또는 플레이어를 시뮬레이션하는 봇 등이 있습니다. 게임 월드의 일부로 존재하는 논플레이어 AI의 경우 플레이어 스테이트가 없습니다.

디폴트 스테이트 : 이름, 현재 레벨, 체력, 점수, 깃발 뺏기
깃발 뺏기 : 깃발 뺏기 게임에서 현재 깃발을 들고 있는지 여부

13. 게임 모드
게임 모드는 플레이 중인 게임의 규칙을 설정합니다. 규칙에는 다음이 포함될 수 있습니다.

플레이어가 게임에 참여하는 방법, 게임의 일시 정지 가능 여부, 승리 조건 등 특정 게임 전용 행동,
프로젝트 세팅에서 디폴트 게임 모드를 설정하고 레벨별로 오버라이드할 수 있습니다. 어떤 방법으로 구현하든, 한 레벨에는 하나의 게임 모드만 존재할 수 있습니다.

14. 게임 스테이트
게임 스테이트는 게임 내의 모든 클라이언트에 복제할 정보가 들어 있는 컨테이너입니다. 쉽게 말해 연결된 모든 사람에 대한 '게임의 스테이트'를 뜻합니다.

게임 스테이트의 몇 가지 예시 : 게임 점수에 대한 정보, 대결이 시작됐는지 여부, 월드에 있는 플레이어 수를 기준으로 스폰할 AI 캐릭터의 수

15. 브러시
브러시 는 큐브, 구체와 같은 3D 셰이프를 묘사하는 액터입니다. 레벨에 브러시를 배치하여 레벨 지오메트리를 정의할 수 있습니다. 이를 바이너리 스페이스 파티션 또는 BSP 브러시라고 합니다. 레벨의 윤곽 작업을 빠르게 하고 싶은 경우 등에 유용합니다.

16. 볼륨
볼륨 은 연결된 효과에 따라 용도가 달라지는 바운드된 3D 공간입니다. 예를 들면 다음과 같습니다.

블로킹 볼륨 - 보이지 않으며 액터가 통과하지 못하게 막습니다.
페인 코징 볼륨 - 오버랩되는 액터에 시간이 지남에 따라 대미지를 줍니다.
트리거 볼륨 - 액터가 들어오거나 나갈 때 이벤트를 유발하도록 프로그래밍됩니다.

17. 레벨
레벨은 여러분이 정의하는 게임플레이 영역입니다. 레벨에는 지오메트리, 폰, 액터 등과 같이 플레이어가 보고 상호작용할 수 있는 모든 것이 포함됩니다.

18. 월드
월드 는 게임을 구성하는 모든 레벨이 담겨 있는 컨테이너입니다. 월드는 레벨의 스트리밍과 다이내믹 액터의 스폰(생성)을 처리합니다.

툴과 에디터

언리얼 엔진 5 는 게임 또는 애플리케이션을 만드는 데 사용할 수 있는 툴 , 에디터 , 시스템의 조합을 제공합니다.
툴- 액터를 레벨에 배치하거나 지형을 페인팅하는 등 특정 작업을 수행하기 위해 사용합니다.
에디터 - 보다 복잡한 결과를 달성하기 위해 사용하는 툴 컬렉션입니다. 예를 들어 레벨 에디터 에서는 게임의 레벨을 빌드할 수 있고, 머티리얼 에디터 에서는 머티리얼의 형태와 느낌을 바꿀 수 있습니다.
시스템 - 게임이나 애플리케이션의 일부 측면을 제작하기 위해 함께 작동하는 대규모의 기능 컬렉션입니다. 예를 들어 블루프린트는 게임플레이 엘리먼트를 시각적으로 스크립팅하는 데 사용되는 시스템입니다.

1. 레벨 에디터
게임플레이 레벨
레벨 에디터 는 게임플레이 레벨을 생성하는 주요 에디터입니다. 여기에 다양한 타입의 액터 및 지오메트리, 블루프린트 비주얼 스크립팅, 나이아가라 등을 추가하여 게임의 플레이 공간을 정의할 수 있습니다. 언리얼 엔진 5는 프로젝트를 생성하거나 열 때 기본적으로 레벨 에디터를 엽니다.

2. 스태틱 메시 에디터
스태틱 메시 에디터를 사용하면 스태틱 메시의 형태, 콜리전, UV 매핑을 프리뷰하고, 프로퍼티를 설정하고 조작할 수 있습니다. 스태틱 메시 에디터에서는 스태틱 메시 에셋의 LOD, 즉 레벨 오브 디테일 세팅을 설정하여 게임이 실행되는 방식과 위치에 따라 해당 에셋을 얼마나 단순하게 또는 복잡하게 나타낼지를 제어할 수 있습니다.

3. 머티리얼 에디터
머티리얼
머티리얼 에디터는 머티리얼을 생성하고 편집하는 곳입니다. 머티리얼은 메시에 적용하여 메시의 시각적 형태를 제어할 수 있는 에셋입니다. 예를 들어 흙 머티리얼을 만들어 레벨의 바닥에 적용하면 흙으로 덮인 것처럼 보이는 표면을 만들 수 있습니다.

4. 블루프린트 에디터
블루프린트
블루프린트 에디터 에서는 블루프린트를 작업하고 수정할 수 있습니다. 블루프린트는 게임플레이 엘리먼트를 생성하거, 머티리얼을 수정하거나, C++ 코드를 작성하지 않고도 기타 언리얼 엔진 기능을 구현할 수 있는 특별한 에셋입니다.

5. 피직스 에셋 에디터
피직스
피직스 에셋 에디터를 사용하면 스켈레탈 메시와 함께 사용할 피직스 에셋을 생성할 수 있습니다. 여기서 실제로 디포메이션과 콜리전 등의 피직스 기능을 구현하게 됩니다. 아무것도 없는 상태에서 시작해 완전한 래그돌 설정을 빌드하거나, 자동화 툴을 사용하여 기본적인 피직스 바디 및 피직스 컨스트레인트 세트를 생성할 수 있습니다.

6. 비헤이비어 트리 에디터
AI 행동
비헤이비어 트리 에디터는 레벨 내 액터에 대해 비주얼 노드 기반의 시스템으로 인공지능을 스크립팅할 수 있는 곳입니다. 적, NPC, 비히클에 대한 다양한 비헤이비어를 원하는 수만큼 생성할 수 있습니다.

7. 나이아가라 에디터
파티클 효과
나이아가라 에디터에서는 각 이펙트에 대한 별도의 파티클 이미터로 구성된 완전 모듈형 파티클 이펙트 시스템을 활용하여 특수 이펙트를 만듭니다. 이미터는 나중에 사용할 수 있도록 콘텐츠 브라우저에 저장해 두고 현재 또는 향후 프로젝트에서 새 이미터의 기초로 사용할 수 있습니다.

8. UMG UI 에디터
유저 인터페이스
UMG(Unreal Motion Graphics) UI 에디터 는 인게임 헤드업 디스플레이, 메뉴, 기타 인터페이스 관련 그래픽과 같은 UI 엘리먼트를 만드는 데 사용할 수 있는 시각적 UI 제작 툴입니다.

9. 폰트 에디터
폰트 에디터를 사용하여 폰트 에셋을 추가, 정리, 프리뷰합니다. 여기서는 폰트 에셋 레이아웃 및 힌팅 정책과 같은 폰트 파라미터를 정의할 수도 있습니다.
폰트 힌팅(Font Hinting) - 모든 크기의 디스플레이에서 텍스트의 가독성을 높이는 수학적인 방법

10. 시퀀서 에디터
시네마틱 및 다이내믹 이벤트

시퀀서 에디터는 인게임 시네마틱을 제작할 수 있는 특수한 멀티트랙 에디터입니다. 레벨 시퀀스를 생성하고 트랙을 추가하여 씬의 콘텐츠를 결정하는 각 트랙의 구성을 정의할 수 있습니다. 트랙은 애니메이션, 트랜스포메이션, 오디오 등으로 이루어질 수 있습니다.

11. 애니메이션 에디터
언리얼 엔진 5의 애니메이션 에디터는 스켈레톤 에셋, 스켈레탈 메시, 애니메이션 블루프린트, 그 외 다양한 애니메이션 에셋을 편집하는 데 사용합니다.

12. 컨트롤 릭 에디터
컨트롤 릭은 엔진 내에서 캐릭터를 직접 리깅할 수 있는 애니메이션 툴 모음입니다. 컨트롤 릭을 사용하면 리깅 및 애니메이팅 작업에 외부 툴을 이용할 필요가 없고, 언리얼 에디터에서 직접 애니메이팅할 수 있습니다. 이 시스템을 사용하면 캐릭터에 대한 커스텀 컨트롤을 생성하고 리깅할 수 있고, 시퀀서에서 애니메이팅할 수 있으며, 애니메이팅 프로세스에 유용한 여러 기타 애니메이션 툴을 사용할 수 있습니다.

13. 사운드 큐 에디터
사운드 큐
언리얼 엔진 5에서의 오디오 재생 행동은 사운드 큐 내에서 정의되며, 사운드 큐 에디터를 사용하여 편집할 수 있습니다. 이 에디터 안에서는 몇 가지 사운드 에셋을 조합하고 혼합하여 사운드 큐로 저장된 하나의 혼합된 출력을 생성할 수 있습니다.

14. 미디어 에디터
외부 미디어 재생
미디어 에디터 를 사용하여 언리얼 엔진 5 내에서 재생하기 위한 소스 미디어로 사용할 미디어 파일이나 URL을 정의합니다.

여기서는 자동 재생, 재생 속도, 루핑과 같은 소스 미디어 재생 방법에 대한 설정을 정의할 수 있지만, 미디어를 직접 편집할 수는 없습니다.

15. nDisplay 3D 환경설정 에디터
버추얼 프로덕션 및 현장 공연
nDisplay는 파워월, 돔, 곡면 화면 같은 여러 동기화된 디스플레이 장치에서 언리얼 엔진 씬을 렌더링합니다. nDisplay 3D 환경설정 에디터(nDisplay 3D Config Editor) 를 사용하면 nDisplay 설정을 생성하고 모든 디스플레이 장치에서 콘텐츠가 어떻게 렌더링될지 시각화할 수 있습니다.

16. DMX 라이브러리 에디터
DMX는 라이브 이벤트 산업 전반에 걸쳐 조명 기구, 레이저, 연기 발생기, 기계 장치, 전자 광고판 같은 다양한 장치를 제어하는 데 사용되는 디지털 통신 표준입니다. DMX 라이브러리 에디터에서는 이들 장치와 필요한 명령을 커스터마이징할 수 있습니다.

좌표계 용어집

a. 스페이스
1. Tangent
직교하는 것으로, 왼편일 수도 오른편일 수도 있습니다. TangentToLocal 트랜스폼에는 로테이션만 있으므로, OrthoNormal 입니다.

2. Local - 오브젝트 씬
직교하는 것으로, 왼편일 수도 오른편일 수도 있/습니다 (트라이앵글 컬링 순서의 조절이 필요합니다). LocalToWorld 트랜스폼에는 로테이션, 비균일 스케일(- 와인딩 순서를 바꿔버릴 수도 있는 음수 비균일 스케일 포함), 트랜슬레이션이 들어 있습니다.

3. World
WorldToView 트랜스폼에는 로테이션과 트랜슬레이션만 들어있어, 뷰 스페이스의 거리는 월드 스페이스와 같습니다.

4. TranslatedWorld
트랜슬레이션 적용 행렬은 복합 트랜스폼 행렬에서 카메라 위치를 제거하는 데 사용되는 것으로, 버텍스 트랜스폼 작업시 정밀도를 향상시킵니다.

5. View - CameraSpace
ViewToClip 트랜스폼에는 X 와 Y 의 스케일은 있지만 트랜슬레이션은 없(어서 중심을 벗어나는 투사 되겠)습니다. Z 에는 스케일과 트랜슬레이션을 적용합니다. 동질성 클립 스페이스로의 변환을 위해 투사를 적용하기도 합니다.

6. Clip- HomogeniousCoordinates, PostProjectionSpace, ProjectionSpace
원근 투사 행렬의 적용 이후입니다. 참고로 클립 스페이스의 W 는 뷰 스페이스의 Z 와 같습니다.

7. Screen - OpenGL(NormalizedDeviceCoordinates)
원근 분할 이후 : 좌/우 -1,1 || 상/하 1,-1 || 근/원 0,1

8. Viewport - ViewportCoordinates, WindowCoordinates
픽셀 단위로 : 좌/우 0, 폭-1 || 상/하 0,높이-1

b. 좌표계 환전
좌표계 사이의 변환(transform)은 항상 X To Y 형태의 이름이 붙습니다.
예: WorldToView, TranslatedWorldToViewTangentToWorld

디렉터리 구조

최상위 레벨에는 Engine 디렉토리와 게임 프로젝트가 있습니다. Engine 디렉토리에는 엔진 자체와, 엔진에서 기본 제공하는 모든 툴이 포함되어 있습니다. 각 게임 폴더에는 해당 게임과 관련된 모든 파일이 들어 있습니다. 이전 버전의 엔진보다 UE4에서 엔진과 게임 프로젝트 간의 분리가 훨씬 더 큽니다.

1. 루트 디렉터리
Engine - 엔진을 이루는 모든 소스 코드, 콘텐츠 등이 포함되어 있습니다.
Templates - 새 프로젝트 생성 시 사용 가능한 프로젝트 템플릿 모음입니다.
GenerateProjectFiles.bat - Visual Studio 에서 엔진과 게임 작업시 필요한 UE4 솔루션 및 프로젝트 파일 생성시 사용됩니다.
UE4Games.uprojectdirs - 엔진이 하위 디렉터리의 프로젝트를 발견할 수 있도록 해주는 헬퍼 파일입니다.

2. 공통 디렉터리
Engine 과 게임 프로젝트 디렉터리 사이에 공통으로 쓰이는 서브디렉터리가 있습니다.

1) Binaries - 컴파일 도중 생성되는 실행 파일이나 기타 파일이 포함됩니다.

2) Build - 엔진이나 게임을 빌드하는 데 필요한 파일은 물론, 플랫폼별 빌드를 만드는 데 필요한 파일도 들어 있습니다.

3) Config - 엔진 행위를 제어하는 값 설정용 환경설정 파일입니다. 게임 프로젝트 Config 파일에 설정된 값은 Engine\Config 디렉터리에 설정된 값을 덮어씁니다.

4) Content - Engine 이나 게임에 대한 콘텐츠, 애셋 패키지와 맵이 들어갑니다.

5) DerivedDataCache - 참조된 콘텐츠에 대해 로드시 생성된 파생 데이터 파일이 들어있습니다. 참조된 콘텐츠에 대해 캐시 파일이 존재하지 않으면 로드 시간이 엄청나게 길어질 수 있습니다.

6) Intermediate - 엔진이나 게임 빌드 도중 생성된 임시 파일이 들어 있습니다. 게임 디렉터리에서 셰이더는 Intermediate 디렉터리에 저장됩니다.

7) Saved - 자동저장, 환경설정 (.ini), 로그 파일이 들어 있습니다. 추가적으로 Engine > Saved 디렉터리에는 크래시 로그, 하드웨어 정보, 스웜 옵션 및 데이터가 들어 있습니다.

8) Source - Engine 이나 게임은 물론 엔진 소스 코드, 툴, 게임플레이 클래스 등에 대한 모든 소스 파일이 들어갑니다.
  a) Engine - Engine 디렉터리에 있는 소스파일은 다음과 다음과 같음
    I. Developer - 에디터와 엔진 둘 다에 쓰이는 파일입니다.
    II. Editor - 에디터에만 쓰이는 파일입니다.
    II. Programs - 엔진이나 에디터에 쓰이는 외부 툴입니다.
    Iv. Runtime - 엔진에만 쓰이는 파일입니다.
  b) Game - 게임 프로젝트 디렉터리의 소스 파일은 모듈별, 모듈 하나 당 디렉터리 하나
    I. Classes - 모든 게임플레이 클래스 헤더 (.h) 파일이 들어갑니다.
    II. Private - 모든 파일과 게임플레이 클래스 구현 파일, 모듈 구현 파일이 들어갑니다.
    III. Public - 모듈 헤더 파일이 들어갑니다.

3. Engine 전용 디렉터리
Engine 디렉터리에만 쓰이는 서브디렉터리가 있습니다.

1) Documentation - 엔진 문서 소스와 퍼블리시된 파일도 들어 있습니다.
  a) HTML - 퍼블리시된 HTML 문서 파일입니다.
  b) Source - 소스 마크다운 문서 파일입니다.

2) Extras - 부가 헬퍼, 유틸리티 파일입니다.

3) Plugins - 엔진에 사용되는 플러그인이 들어있습니다.

4) Programs- UE4 루트 디렉터리에 저장된 프로젝트와UnrealFrontend, UnrealHeaderTool 등의 기타 언리얼 프로그램용 환경설정 파일과 로그 파일이 들어 있습니다.

5) Shaders - 엔진에 대한 셰이더 소스 (.usf) 파일이 들어갑니다.

4. 게임 프로젝트 디렉터리
1) Binaries 컴파일 도중 생성된 실행 파일이나 기타 파일이 들어 있습니다.

2) Config - 게임의 기본 프로젝트 세팅입니다.

3) Content - 엔진이나 게임의 콘텐츠는 물론 애셋 패키지와 맵까지도 들어 있습니다.

4) External dependencies - 공용 Rocket 엔진 헤더 파일을 표시합니다.

5) Intermediate - Visual Studio 프로젝트 파일처럼 UnrealBuildTool 이 생성하는 파일이 들어갑니다. 이 파일들은 지우고 재생성 가능합니다.

6) Saved - 컨픽 파일과 로그 등 엔진이 생성하는 파일이 들어갑니다. 이 파일들은 지우고 재생성 가능합니다.

7) Source - 게임 모듈 오브젝트 클래스 파일이 들어갑니다.

5. 솔루션 디렉터리
1) Classes - 게임 오브젝트 클래스 정의 (.h) 파일이 들어갑니다.

2) Config - 게임의 기본 프로젝트 세팅입니다.

3) External dependencies - 공용 Rocket 엔진 헤더 파일을 표시합니다.

4) Private - 개인 게임 오브젝트 클래스 구현 (.cpp) 파일이 들어갑니다.

5) Public - 공용 게임 오브젝트 클래스 구현 (.cpp) 파일이 들어갑니다.