

# DAY 26

## ▼ 관련 문서

스크랩: <https://velog.io/@cedongne/UE5-Unreal-Engine-5-길라잡이-7.-게임-모드>

AGameModeBase:

<https://docs.unrealengine.com/4.27/ko/InteractiveExperiences/Framework/GameMode/>

게임플레이 프레임 워크:

<https://docs.unrealengine.com/4.27/ko/InteractiveExperiences/Framework/>

UMG, HUD, Slate: <https://hwan1402.tistory.com/104>

유저 인터페이스와 HUD:

<https://docs.unrealengine.com/4.27/ko/InteractiveExperiences/Framework/UIAndHUD/>

언리얼 엔진의 게임플레이 프레임워크는 미리 완성해둔 레벨에 규칙을 부여하고 레벨에서 활동할 액터를 배치한다.

이때 규칙을 '게임 모드', 배치한 액터 중 플레이어가 직접 조종할 수 있는 액터를 '폰'이라고 한다

## 게임 모드

- 게임의 규칙을 결정하고 정보를 처리하는 액터



### 게임 모드가 결정하는 규칙의 예

- 플레이어가 게임에 들어오는 방식, 스폰 위치 선택 규칙
- 기타 스폰/리스폰 동작 포함 가능
- 액터간의 상호작용 처리
- 존재하는 플레이어와 관람자의 수는 물론, 허용된 플레이어와 관람자 최대 수
- 게임 일시정지 가능 여부, 게임 일시정지 처리 방식
- 레벨 간의 전환, 게임의 시네마틱 모드 시작 여부 포함

→ 이러한 게임모드에서 관리하는 규칙의 특징은 **게임 중에 변하지 않을 정보들이라는 것**

\*\* 자주 변경될 수 있는 정보는 게임 모드에서 관리하지 않는 것이 좋다

\*\* 또한 게임 형식 미션 유형, 특별 지역 규칙 등에 따라 AGameModeBase의 서브 클래스를 생성할 수 있다

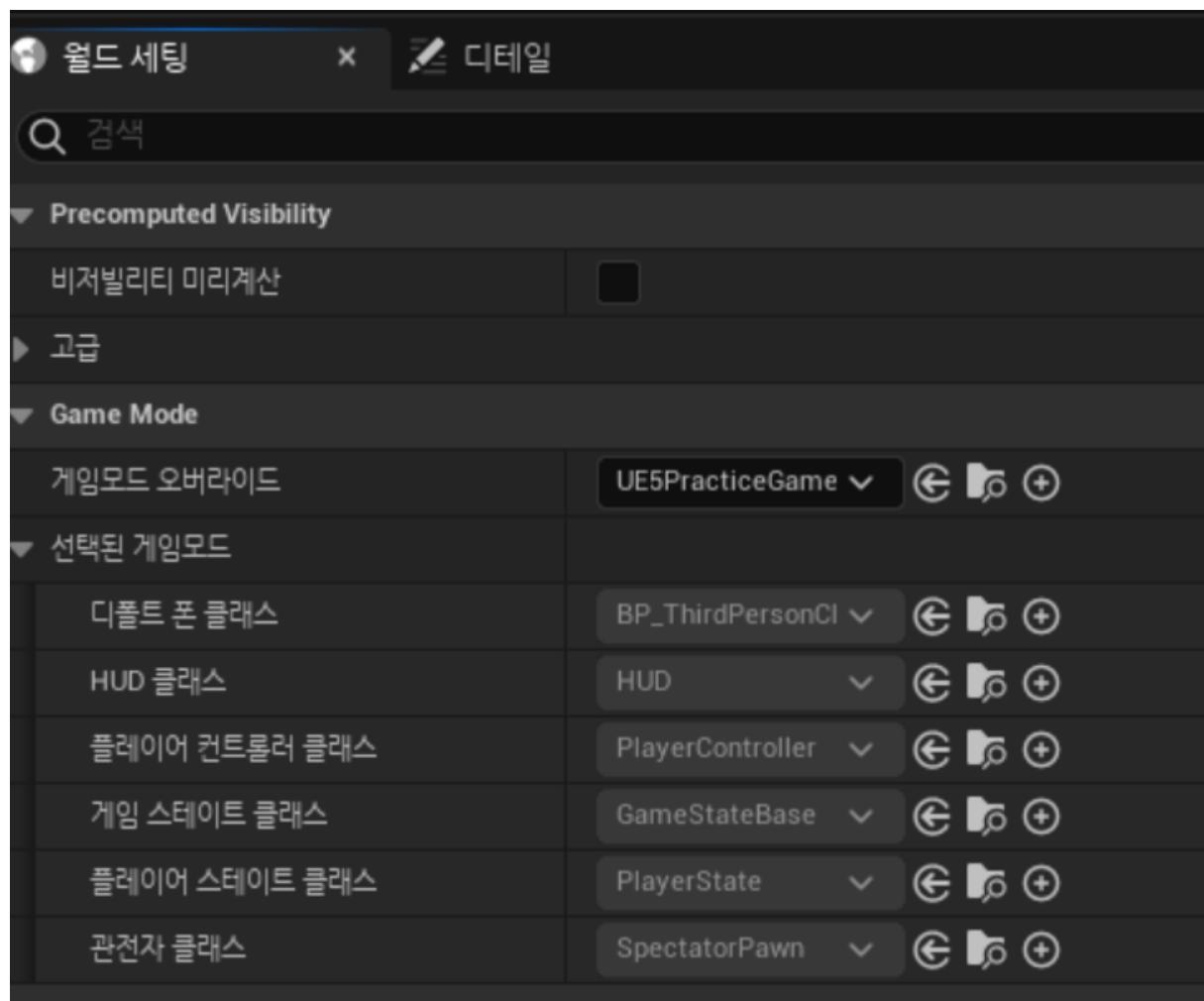
\*\* 한 게임에 여러 게임 모드가 있을 수 있지만, 한 번에 하나의 게임모드만 사용할 수 있다

## 현재 프로젝트가 어떤 게임 모드를 따르고 있는지 알아보는 방법

- 상단 편집 - 프로젝트 세팅 - 프로젝트 - 맵 & 모드에서 확인 가능
- 기본적으로 'GameModeBase'가 선택되어 있으며, 사용자가 직접 게임 모드를 지정해줄 수 있다
- 또한 선택된 게임 모드에 따라 게임을 구성하는 다양한 클래스를 커스텀하여 지정해줄 수 있다

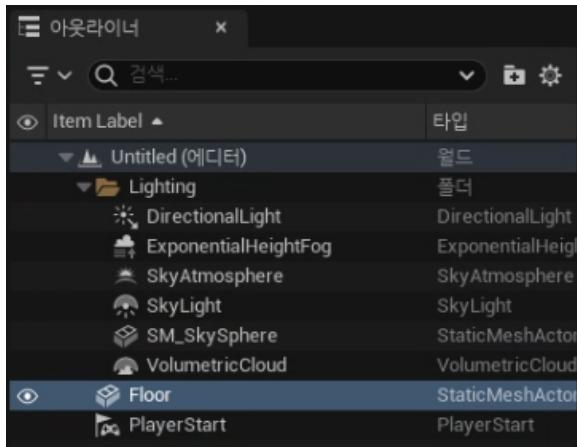
## 게임 모드 오버라이딩

- 프로젝트 세팅 창에서 설정한 게임 모드는 모든 레벨에 기본으로 적용되지만, 특정 레벨에만 다른 게임 모드를 적용하고 싶다면 해당 레벨이 열린 상태에서 상단 창 - 월드 세팅 패널을 열어 게임 모드를 오버라이딩 해줄 수 있다

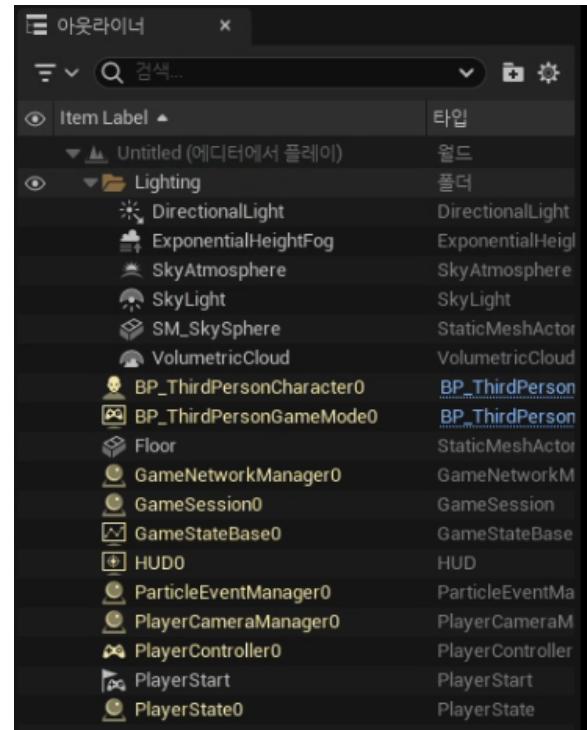


## 실습

1. Ctrl + N을 눌러 Basic 레벨을 하나 만든다
2. 플레이를 하면 아웃라이너에 플레이하기 전보다 더 추가된 것들을 볼 수 있다



Play 전 아웃라이너 상태



Play시 아웃라이너 상태

→ 게임을 플레이할 때 기본적으로 필요로 하는 것들을 자동적으로 생성해주는 것을 확인할 수 있다



**월드세팅 패널의 Game Mode - 게임모드 오버라이드는 None으로 되어있는데 왜 생성되는 걸까?**

- 상단 편집 - 프로젝트 설정 클릭
- 프로젝트 창의 맵 & 모드 - 기본 게임모드와 선택된 게임모드를 보면 기본적으로 설정되어 있는 것을 확인할 수 있다



→ 이렇게 기본적으로 설정되어 있기 때문에 게임모드 오버라이드가 None이여도 플레이시 생성되는 것이다

## 각 클래스의 역할

### 디폴트 폰 클래스

- Pawn(폰) : 컨트롤러로 조종가능한 액터
- 게임 플레이모드에 진입하면 디폴트 폰 클래스로 설정된 폰이 스폰되고, 컨트롤러에 빙의(Possess)된다  
→ 이는 플레이어가 직접 조종하는 플레이어 컨트롤러일 수도 있고, 컴퓨터가 조종하는 AI 컨트롤러일 수도 있다

### 플레이어 컨트롤러 클래스

- 컨트롤러(Controller) : 폰, 캐릭터를 직접 제어하는 클래스
  - 하나의 컨트롤러는 하나의 폰만 조종할 수 있기 때문에 폰에 빙의(Possess)된다고 한다
  - 플레이어가 직접 조종하는 플레이어 컨트롤러, 컴퓨터가 조종하는 AI 컨트롤러로 구분된다
  - 제어 중인 폰에 발생하는 이벤트를 가로채 폰의 기본 반응을 대체하는 반응을 설정할 수 있다
  - 전략 시뮬레이션과 같이 게임 유형에 따라 한 컨트롤러가 여러 폰을 제어해야하는 상황이 있을 수도 있는데 이는 약간의 조정으로 해결이 가능하다
  - 상황에 따라 컨트롤러는 폰을 옮겨다닐 수 있지만, 플레이어는 자신이 배정 받은 플레이어 컨트롤러를 변경할 수 없다



플레이어 컨트롤러의 경우 플레이어의 입력을 폰이 직접 받는 것이 아니라, 플레이어 컨트롤러가 직접 입력을 받고 폰에게 적절한 행동을 하도록 명령을 내리는 것이다

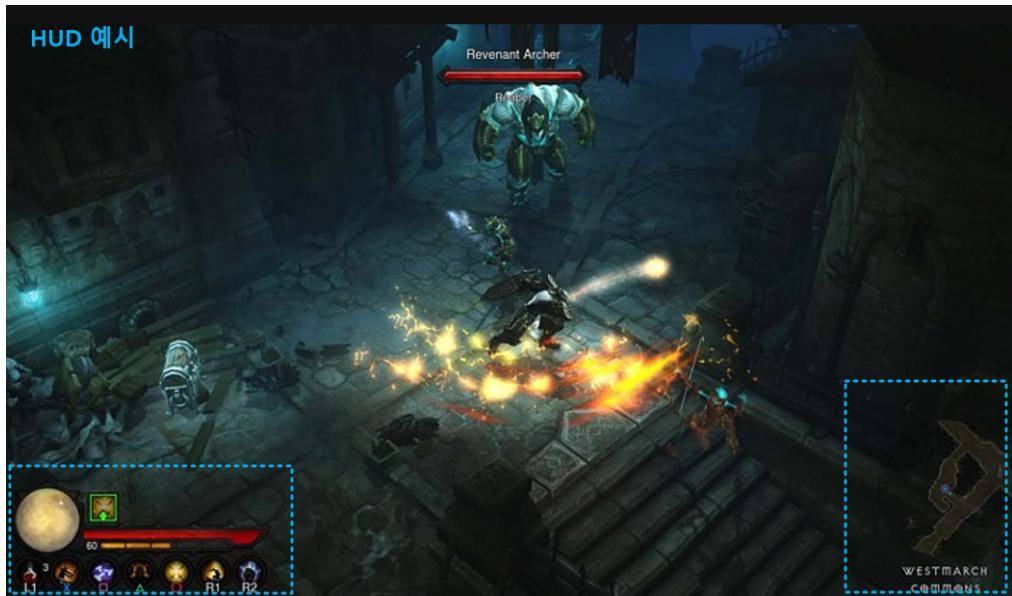
AI 컨트롤러도 같은 방식으로 정해진 로직에 따라 폰이 아닌 AI 컨트롤러에게 해야 할 행동을 알려주면 컨트롤러가 폰을 직접 움직인다

- 플레이어 컨트롤러는 제오할 폰 클래스에 빙의되고 HUD, 입력 컨트롤, PlayerCameraManager를 설정한다

### HUD 클래스

- HUD
  - Head-Up Display의 약자
  - 원래 비행기나 자동차 앞 유리에 정보를 표시해주는 증강현실 장치를 칭하는 용어

- 게임에서 플레이어의 상태나 정보를 화면 위에 겹쳐놓이는 기본적인 인터페이스를 뜻하기도 한다



양끝 하단의 에너지, 미니맵, 아이템 등의 인터페이스 요소와 같은 것들을 의미

- Head-Up-Display의 기초 클래스로, AHUD라는 클래스명으로 정의되어 있다



### UMG(Unreal Motion Graphic)

- 언리얼에서는 상호작용이 불가능한(클릭할 수 없는) 인터페이스 요소등만 HUD라고 하지만, UE4부터 등장한 UMG(Unreal Motion Graphic)이라는 UI 제작 툴이 개발된 이후로 상호작용 가능한 일부 요소도 HUD에 포함시킨다
- UE4부터 등장한 UMG 툴이 훨씬 많은 장점을 갖고 있어 HUD 클래스보다 UMG의 사용이 권장되고 있다
- 그럼에도 아직 HUD 클래스가 남아 있는 이유는 이전 버전 시스템과 호환 즉, 레거시 때문이다
- HUD 클래스와 UMG는 완전히 독립되어 수행되는 UI이므로 둘을 동시에 사용한다고 충돌이 일어나지는 않는다

## 게임 스테이트 클래스

- 게임 스테이트(Game State)
  - 게임에 규칙 관련 이벤트가 발생하고 트래킹을 통해 정보를 모든 플레이어가 공유할 필요가 있을 때, 그 정보는 게임 스테이트에 보관되고 그를 통해 동기화된다



### 예시

- 게임 실행 기간(로컬 플레이어 참가 전 실행 시간 포함)
  - 접속한 플레이어 목록, 각 플레이어의 게임 참가 기간, 현재 상태
  - 오픈 월드 게임에서 완료한 미션 목록
  - 현재 게임 모드의 베이스 클래스
  - 게임 시작 여부
- 
- 클라이언트가 게임의 상태를 모니터링할 수 있도록 해준다
  - 플레이어가 개개인이 아닌 접속한 모든 클라이언트가 알아야하는 정보, 게임 모드에 관련된 게임 전반적인 프로퍼티 기록을 유지할 수 있다
  - AGameStateBase가 기본 구현이며, 게임에서 무슨 일이 벌어지고 있는지 플레이어에게 지속적으로 알리기 위해, 필요한 변수와 함께 함수를 넣기 위해 C++ 또는 블루프린트로 자주 확장시키는 클래스
  - 게임 모드는 서버에만 존재하지만 게임 스테이트는 서버에 존재하면서 모든 클라이언트에 리플리케이트되어, 연결된 모든 기기의 게임 최신 상태를 유지한다

## | 플레이어 스테이트 클래스

- 플레이어 스테이트(Player State)
  - 플레이어 개인의 정보를 관리하기 위한 클래스
  - 예를 들어 FPS 게임에서 팀 킬 스코어 중 특정 플레이어의 킬 스코어가 몇 점인지와 같은 개별 플레이어의 기록을 이곳에 저장한다



### 게임 스테이트와 플레이어 스테이트

- 게임 스테이트: 게임 플레이 도중 변하면서 '모두에게' 관련이 있고 보일 수 있는 프로퍼티 기록을 유지
- 반면에 플레이어 스테이트는 '개인'의 정보를 관리한다

## | 관전자 클래스

- Spectator Class
- 관람자/관중 클래스라고도 한다

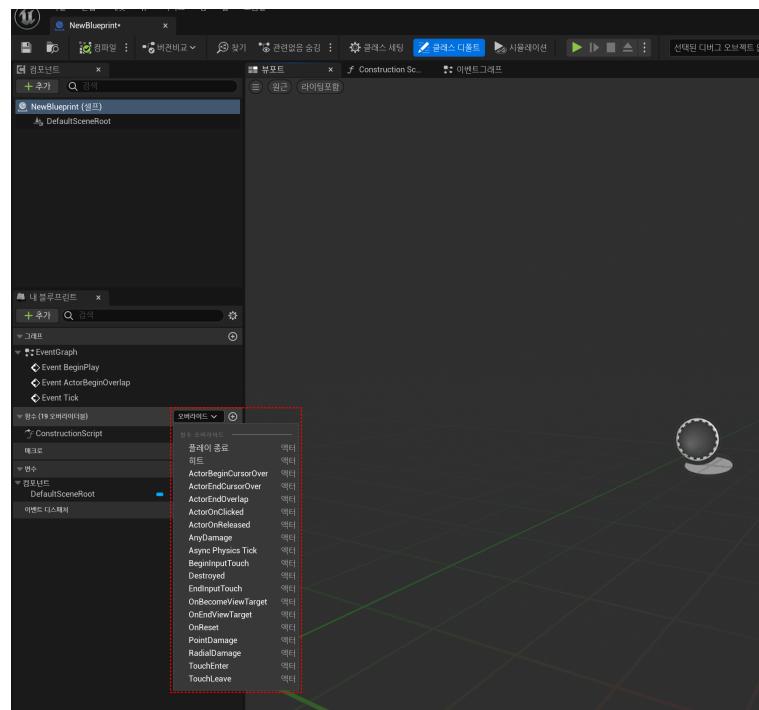
- 플레이어가 게임의 직접 참여자가 아닌 관중의 역할로 있을 때 무중록 비행 동작 등 관람에 이상적인 단순한 프레임 웍크를 제공한다

## 블루프린트

- 블루프린트는 이벤트 기반으로, 이벤트와 액션의 조합으로 이루어져 있다
- 생성하는 방법: 콘텐츠 브라우저에서 우클릭 - 블루프린트 클래스 - 원하는 클래스 선택

## 이벤트

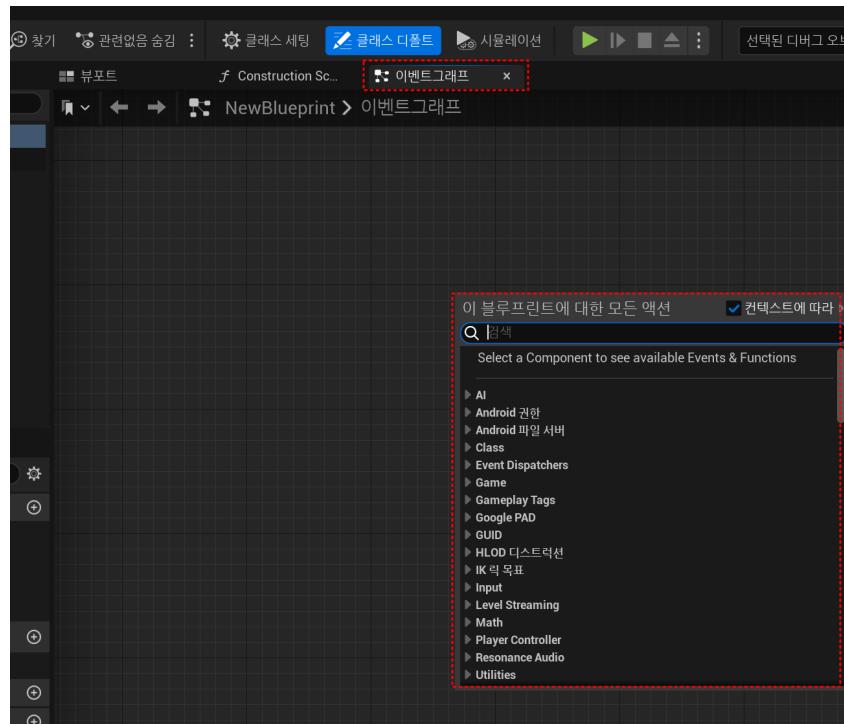
- EventGraph 내의 개별 네트워크 실행을 시작하기 위해 게임 플레이 코드에서 호출되는 노드
- 이를 통해 블루프린트는 게임 시작, 레벨 재설정, 플레이어가 피해를 입을 때 등 게임내에서 발생하는 특정 이벤트에 대응하여 일련의 작업을 수행할 수 있다
- 이벤트는 단일 개체만 실행할 수 있다
- 하나의 이벤트에서 여러 작업을 처리하려면 해당작업을 선형으로 연결해야한다
- 엑터가 제공하는 모든 이벤트를 확인하는 방법: 블루프린트 창 - 좌측 함수 패널 - 오버라이드 클릭



이벤트를 클릭하면 이벤트 그래프 창에 노드가 생성되고 오버라이드 목록에서는 사라진다

## 액션

- 액터가 제공하는 모든 액션을 확인하는 방법: 이벤트 그래프 창의 빈 곳에서 우클릭

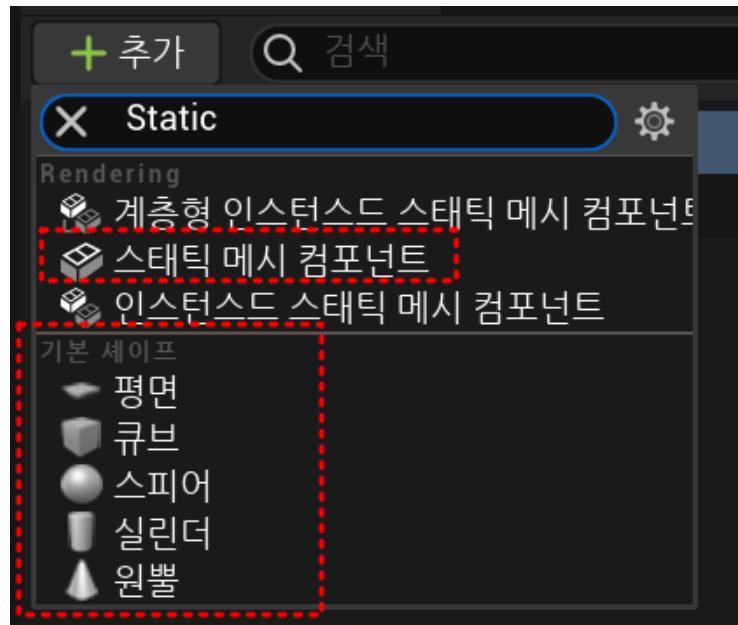


## 컴포넌트

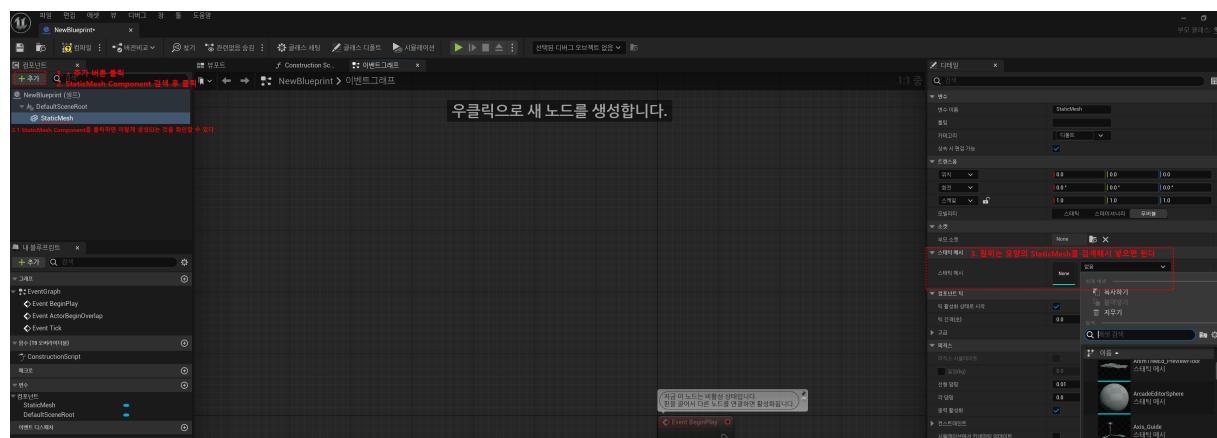
처음 블루프린트를 만들어 레벨에 올려놓고 플레이 했을 때, 액터의 모양이 표시되지 않는 것을 확인할 수 있다

→ 컴포넌트를 이용해 액터의 모양을 생성해주어야 표시가 된다

- 스태틱 컴포넌트 생성: 상단 컴포넌트 패널 + 추가 버튼 클릭, StaticMesh Component 검색

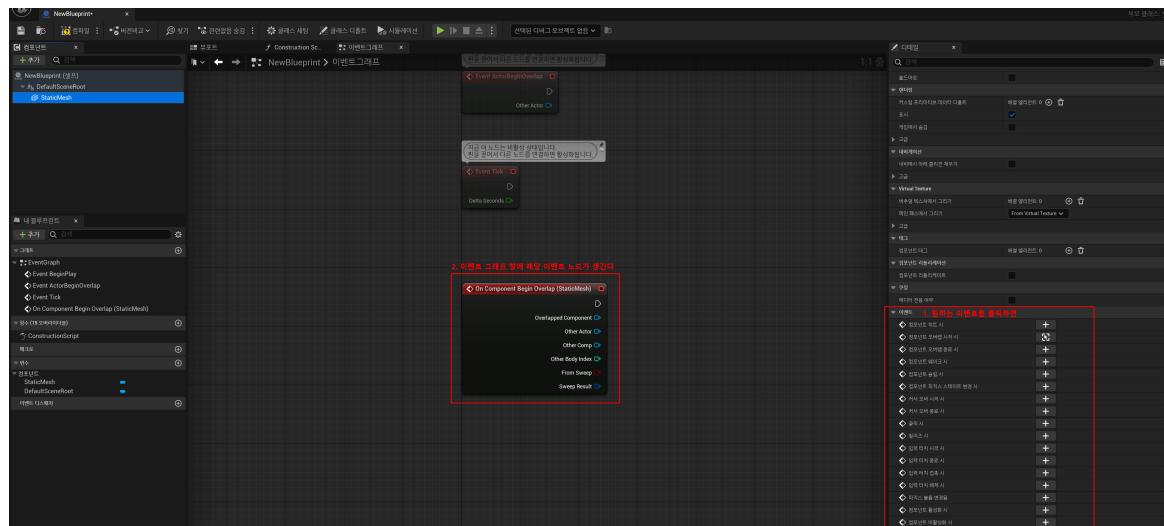


편명, 큐브, 스피어 등도 결과적으로는 스탠티 메시 컴포넌트이다



### • 컴포넌트의 이벤트

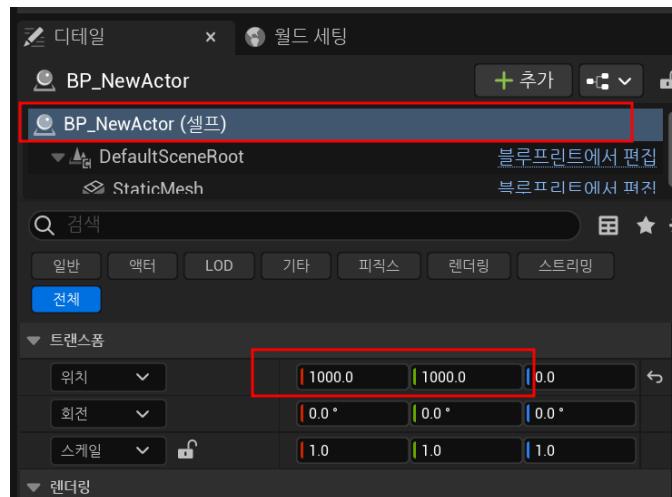
- 컴포넌트에 관한 이벤트를 구현하고 싶다면 좌측 컴포넌트 패널에서 해당 컴포넌트를 클릭하고 우측 디테일 창 하단 이벤트 창에서 원하는 이벤트 클릭



## Location

### 1. World Location

- 절대적인 위치



액터를 레벨에 배치했을 때 해당 액터의 디테일 창에서 확인할 수 있는 위치 값

### 2. Relative Location

- 부모의 위치를 기본값(0, 0, 0)으로 할 때의 상대적인 위치



액터를 부모로 둔 StaticMesh의 위치 값

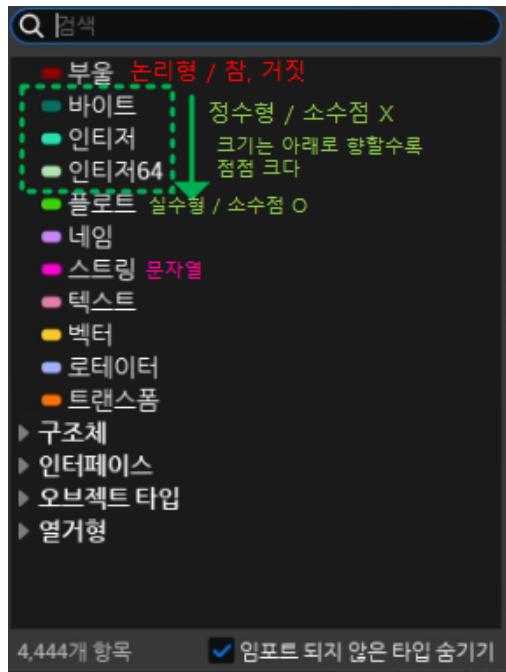


### 3. Local Location

- 액터가 하위에 속해있을 때, 상위 액터의 위치를 디풀트로 하는 상대적 위치

## 변수

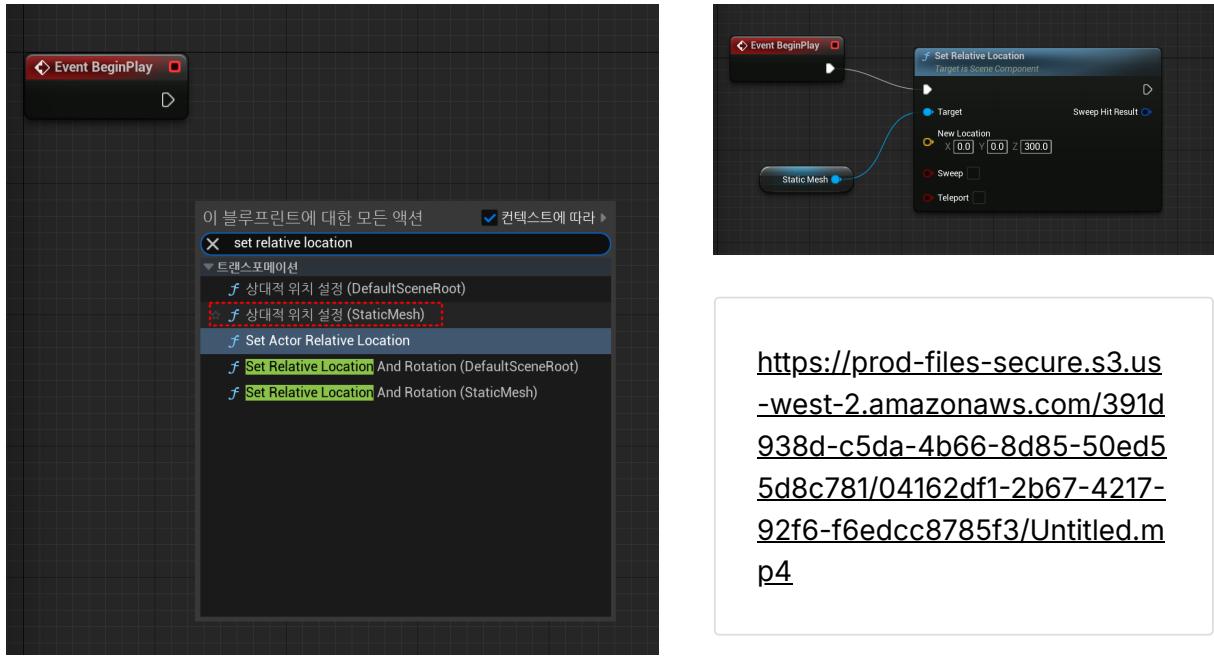
- 자료형에 따라 다른 형태의 데이터를 받을 수 있다



## 실습

### Cube 1회 이동 구현

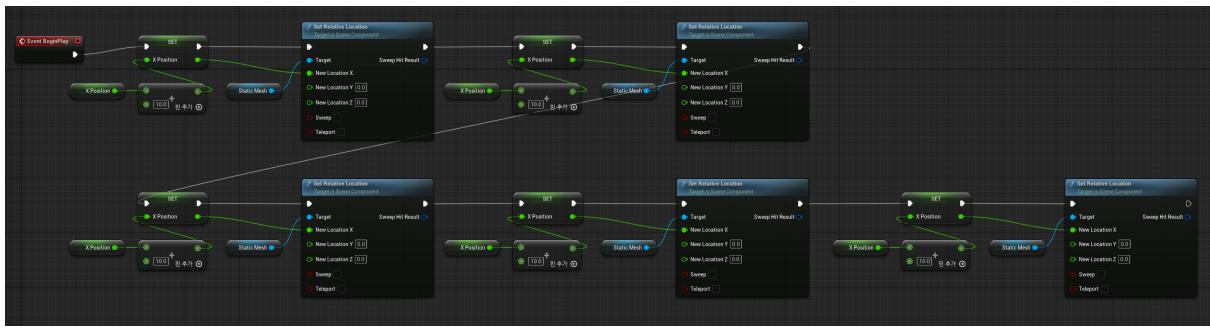
1. 콘텐츠 브라우저에서 우클릭 - 블루프린트 클래스 - 액터 클릭해서 블루 프린트 생성
2. 블루프린트 더블클릭 - 컴포넌트 패널 + 추가 버튼 클릭 - 스태틱 메시 컴포넌트 검색 후 생성
3. 좌측 컴포넌트 패널에서 생성한 스태틱 메시 클릭
4. 우측 디테일 창 - 스태틱 메시에 원하는 스태틱 메시 할당
5. 이벤트 그래프 창에서 'Set Relative Location' 검색 후 생성
  - New Location값 변경
6. Event Begin Play와 Set Relative Location 실행핀 연결



<https://prod-files-secure.s3.us-west-2.amazonaws.com/391d938d-c5da-4b66-8d85-50ed55d8c781/04162df1-2b67-4217-92f6-f6edcc8785f3/Untitled.mp4>

## Cube 5회 이동 구현(10씩 연속해서 움직이기)

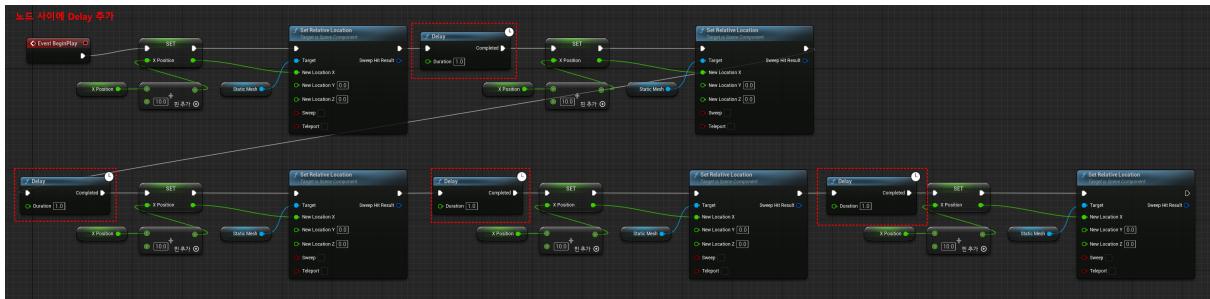
1. 이벤트 그래프 좌측 변수 패널 + 버튼 클릭, 변수 생성
  - 변수이름 - X\_Position, 형 - 플로트
2. X\_Position을 Get으로 이벤트 그래프 창에 끌어다 놓기
  - Get X\_Position의 아웃풋을 끌어서 Add 검색 후 추가
  - Add의 두번째 인풋에 10 입력
3. X\_Position을 Set으로 이벤트 그래프 창에 끌어다 놓기
  - Add의 아웃풋을 Set X\_Position의 X\_Position 인풋에 연결
4. Set Relative의 New Location 인풋 우클릭 - 구조체 편집기 클릭
  - Set X\_Position의 X\_Position 아웃풋을 Set Relative의 New Location X 인풋에 연결
5. 실행선 연결
  - Event Begin Play → Set X\_Position → Set Relative 순으로 연결
6. Get X\_Position, Add, Set X\_Position, Set Relative 노드를 복사해서 순서대로 연결



<https://prod-files-secure.s3.us-west-2.amazonaws.com/391d938d-c5da-4b66-8d85-50ed55d8c781/561c0792-08e3-4734-afb7-d669a8001a3f/Untitled.mp4>

한번에 50을 이동하는 것을 확인할 수 있다

→ 노드 사이에 Delay를 연결하면 10씩 따로 움직이는 것을 볼 수 있다



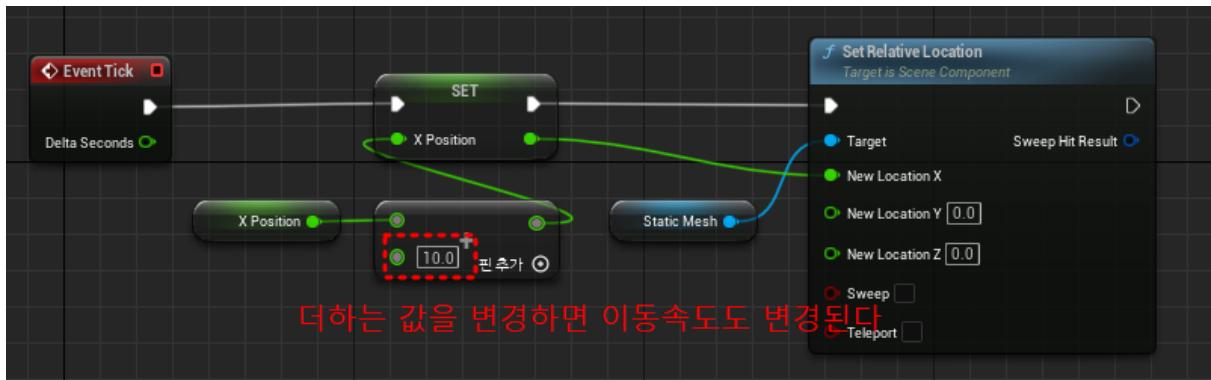
<https://prod-files-secure.s3.us-west-2.amazonaws.com/391d938d-c5da-4b66-8d85-f67821c6-9b01-45a0-9f77-5239999b1dc4/Untitled.mp4>

## Cube 무한 이동

Tick 이벤트를 사용하면 된다

- Tick : 어떤 간격(프레임)에 따라 매번 이벤트를 발생

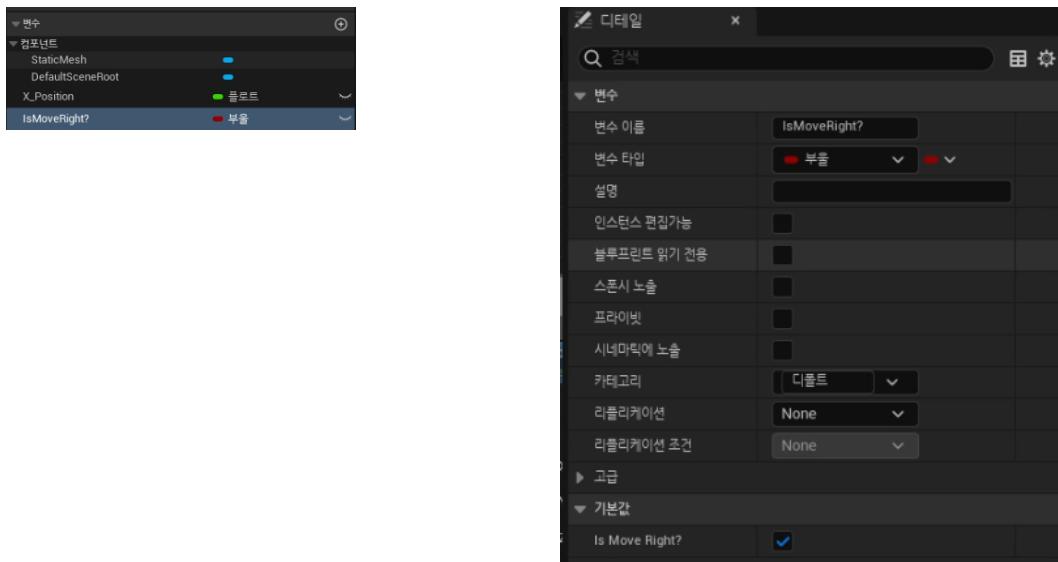
Event Tick의 실행선에 Set X\_position을 연결해주면 끝!



<https://prod-files-secure.s3.us-west-2.amazonaws.com/391d938d-c5da-4b66-8d85-50ed55d8c781/754e9da6-932a-4b91-b26c-9abbd3fa7d02/Untitled.mp4>

## 좌우로 움직이는 큐브 만들기

- 좌측 변수 패널에서 + 버튼 클릭해서 변수 생성
  - 이름: IsMoveRight?, 형: 부울, 기본값: 활성화
  - 생성한 후 Get으로 이벤트 그래프창에 끌어다 놓기



- Event Tick 우측 실행선 끌어서 Branch 검색 후 생성
  - Branch의 Condition 인풋에 Get IsMoveRight? 연결
- 좌측 변수 창에서 X\_Position을 Set, Get으로 하나씩 끌어오기

- Branch의 True 실행선과 Set X\_Position의 좌측 실행선 연결

4. Get\_Position의 아웃풋 끌어서 Add 검색 후 생성

- Add의 두번째 인풋에 1 입력
- Add의 아웃풋을 Set Position의 X\_Position 인풋에 연결

5. 이벤트 그래프창 빈 곳에서 우클릭, Set Relative Location(Static Mesh) 검색 후 생성

- Set Relative Location의 New Location 인풋 우클릭, 구조체 핀 분할
- Set X\_Position의 우측 실행선과 Set Relative Location의 좌측 실행선 연결
- Set X\_Position의 초록색 아웃풋을 Set Relative Location의 New Location X 인풋과 연결

6. Get StaticMesh의 아웃풋을 끌어서 Get Relative Location 검색 후 생성

- Relative Location의 아웃풋 우클릭, 구조체 핀 분할

7. Get Relative Location의 Relative Location X 아웃풋을 끌어서 Greater 검색 후 생성

- 두번째 인풋에 200 입력

8. Set Relative Location의 우측 실행선을 끌어서 Branch 검색 후 생성

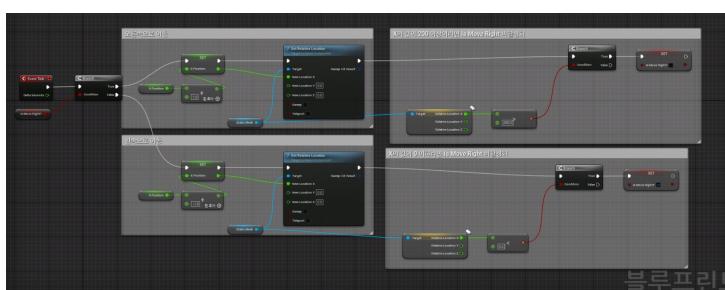
- Branch의 Condition 인풋에 Greater의 아웃풋 연결

9. 좌측 변수 패널에서 IsMoveRight?를 Set으로 가져오기

- Branch의 True 실행선과 Set IsMoveRight?의 좌측 실행선 연결
- Set IsMoveRight?의 IsMoveRight? 인풋 비활성화

10. 첫번째 Branch 이후의 노드들 전부 복사, 붙여넣기

- 복사된 Add의 두번째 인풋 -1로 변경
- 복사된 Greater 노드대신 Less 노드로 변경, Less 노드의 두번째 인풋은 0으로 입력
- 복사된 Set IsMoveRight?의 IsMoveRight? 인풋 활성화

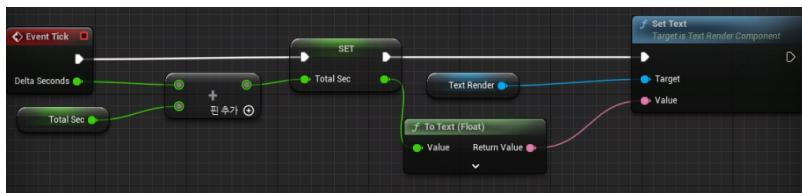


<https://prod-files-secure.s3.us-west-2.amazonaws.com/391d938d-c5da-4b66-8d85-50ed55d8c78/b098226a-f6dc-4106-ae0d-f4f7ef7967fb/Untitled.mp4>

## Timer 제작 실습

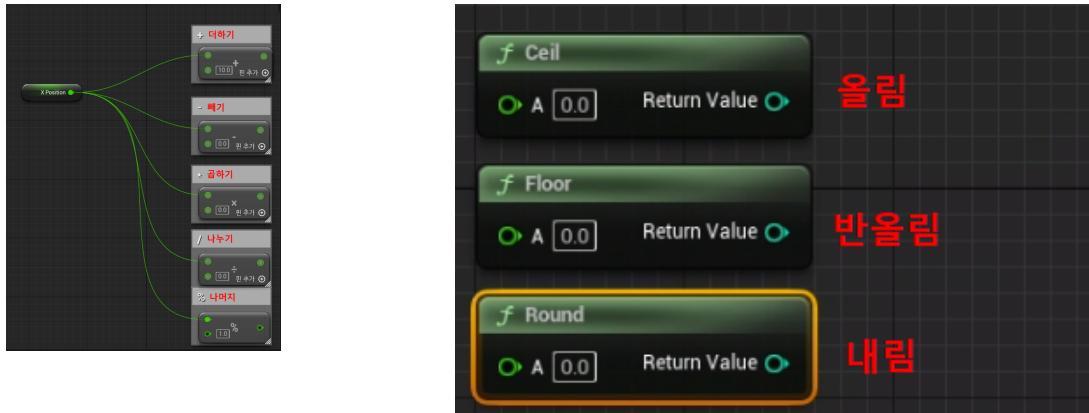
## 베이직

1. 콘텐츠 브라우저 - 우클릭 - 블루프린트 클래스 - 액터 생성
  - 이름은 BP\_Timer
  - 더블클릭해서 블루프린트 에디터 창 들어오기
2. 좌측 컴포넌트 패널 + 추가 버튼 클릭, 텍스트 렌더 컴포넌트 검색 후 생성
  - 좌측 디테일 패널에서 가로정렬 - Center, 세로 정렬 Text Center로 변경
3. 좌측 변수 패널에서 + 버튼 클릭하여 변수 추가
  - 이름: TotalSec / 변수형: 플로트
4. Event Tick의 Delta Seconds의 아웃풋을 끌어서 Add 검색 후 생성
5. 좌측 변수 패널에서 TotalSec을 Get과 Set으로 각각 가져오기
  - Get TotalSec의 아웃풋을 Add의 두번째 인풋에 연결
  - Add의 아웃풋을 Set TotalSec의 TotalSec 인풋에 연결
  - Event Tick과 Set TotalSec의 실행선 연결
6. Set TotalSec의 실행선 끌어서 Set Text(TextRender) 검색 후 생성
  - Set TotalSec의 초록색 아웃풋과 Set Text의 Value 인풋 연결



<https://prod-files-secure.s3.us-west-2.amazonaws.com/391d938d-c5da-4b66-8d85-50ed55d8c781/21dfa73a-65c4-47f6-b1da-858023d52bfc/Untitled.d.mp4>

## 연산자와 올림, 반올림, 내림



## 타이머를 0:0 형식으로 표현

1. Event Tick의 Delta Seconds의 아웃풋을 끌어서 Add 검색 후 생성
2. 좌측 변수 패널에서 TotalSec을 Get과 Set으로 각각 가져오기
  - Get TotalSec의 아웃풋을 Add의 두번째 인풋에 연결
  - Add의 아웃풋을 Set TotalSec의 TotalSec 인풋에 연결
  - Event Tick과 Set TotalSec의 실행선 연결
3. Set TotalSec의 실행선 끌어서 Set Text(TextRender) 검색 후 생성
4. Set TotalSec의 아웃풋 끌어서 Divide 검색 후 생성
  - Divide의 두번째 인풋에 60 입력
5. Divide의 아웃풋 끌어서 Floor 검색 후 생성
6. Set TotalSec의 아웃풋 끌어서 % 검색 후 생성
  - %의 두번째 인풋에 60 입력
7. %의 아웃풋 끌어서 Floor 검색 후 생성
8. 우클릭 Append 검색 후 생성
  - 핀 추가 클릭해서 총 3개의 핀 만들기
  - Divide 연결된 Floor를 Append의 A 인풋에 연결
  - Append의 B 인풋에 : 입력
  - % 연결된 Floor를 Append의 C 인풋에 연결
  - Append의 아웃풋을 Set text의 Value 인풋에 연결



<https://prod-files-secure.s3.us-west-2.amazonaws.com/391d938d-c5da-4b66-8d85-50ed55d8c781/cd4e972e-d58a-49f0-970e-aaed3c73564e/Untitled.mp4>

## 타이머를 00:00 형식으로 표현

1. 좌측 함수 패널 + 버튼 클릭
  - 이름: FloatToStr00
2. 함수 클릭 - 우측 디테일 창에서 입력, 출력 아규먼트 추가
  - 입력: 이름은 Num, 변수형은 플로트
  - 출력: 이름은 Str00, 변수형은 스트링

### Float To Str 00 함수 더블클릭

1. 함수 이벤트 그래프 창에서 Float to Str 00 노드의 실행핀 끌어서 Branch 검색 후 생성
2. Float to Str 00의 Num 아웃풋 끌어서 Less 검색 후 생성
  - Less의 두번째 인풋에 10 입력
  - Less의 아웃풋을 Branch의 Condition에 연결
3. Float to Str 00의 Num 아웃풋 끌어서 Floor 검색 후 생성
4. 이벤트 그래프창 빈 곳에서 Append 검색 후 생성
  - A 인풋에 0 입력
  - B 인풋에 Floor의 Return Value 연결
5. Branch의 True 실행선을 Return Node의 실행선과 연결
  - True가 연결된 Return Node의 Str 00에 Append의 Return Value 연결
6. Return Node 복사 붙여넣기
  - Branch의 False 실행선과 복사된 Return Node의 실행선 연결
  - Floor의 Return Value를 복사된 Return Node의 Str 00과 연결

### 이벤트 그래프 창

1. Event Tick의 Delta Seconds의 아웃풋을 끌어서 Add 검색 후 생성

## 2. 좌측 변수 패널에서 TotalSec을 Get과 Set으로 각각 가져오기

- Get TotalSec의 아웃풋을 Add의 두번째 인풋에 연결
- Add의 아웃풋을 Set TotalSec의 TotalSec 인풋에 연결
- Event Tick과 Set TotalSec의 실행선 연결

## 3. Set TotalSec의 우측 실행선 끌어서 FloatToStr00 검색 후 생성

## 4. Set TotalSec의 초록색 아웃풋 끌어서 Divide 검색 후 생성

- Divide의 두번째 인풋에 60 입력
- Divide의 아웃풋을 FloatToStr00의 Num 인풋에 연결

## 5. Set TotalSec의 초록색 아웃풋 끌어서 % 검색 후 생성

- %의 두번째 인풋에 60 입력

## 6. FloatToStr00의 우측 실행선을 끌어서 FloatToStr00 검색 후 생성

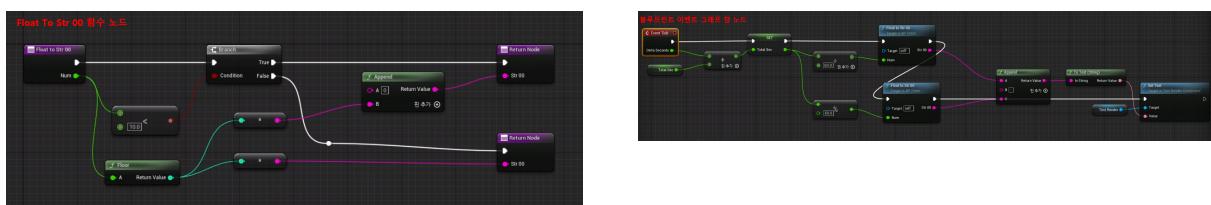
- %의 초록색 아웃풋을 2번째 FloatToStr00의 Num 인풋에 연결

## 7. 우클릭 Append 검색 후 생성

- 핀 추가 해서 총 3개의 핀 만들기
- A 인풋에 1번째 FloatToStr00의 Str00 연결
- B 인풋에 : 입력
- C 인풋에 2번째 FloatToStr00의 Str00 연결

## 8. 두번째 FloatToStr00의 우측 실행선 끌어서 Set Text(Text Render) 검색 후 생성

- Set Text의 Value에 Append의 Return Value 연결

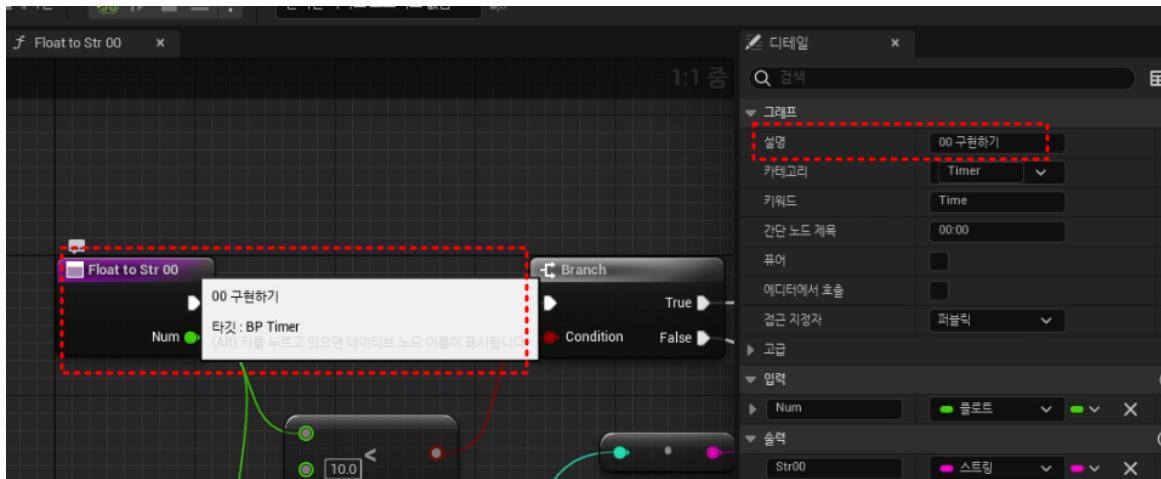


<https://prod-files-secure.s3.us-west-2.amazonaws.com/391d938d-c5da-4b66-8d85-50ed55d8c781/9efcdae9-2298-4762-bc14-c4378ccf2a72/Untitled.mp4>

## 함수의 기능(함수의 디테일 패널)

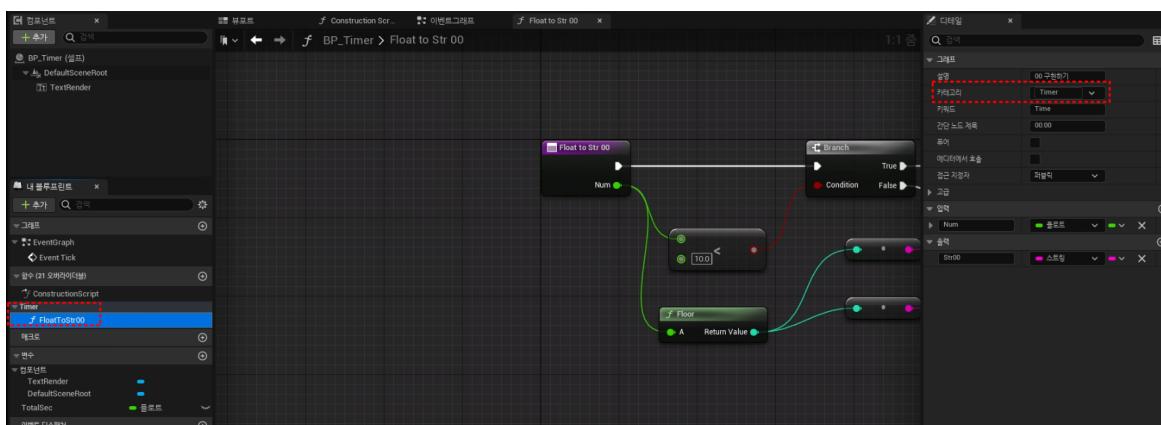
## 설명

- 노드에 마우스 커서를 올리면 설명에 입력했던 내용이 나오는 것을 확인할 수 있다



## 카테고리

- 좌측 함수 패널에서 내가 지정한 카테고리에 함수가 들어가 있는 것을 볼 수 있다



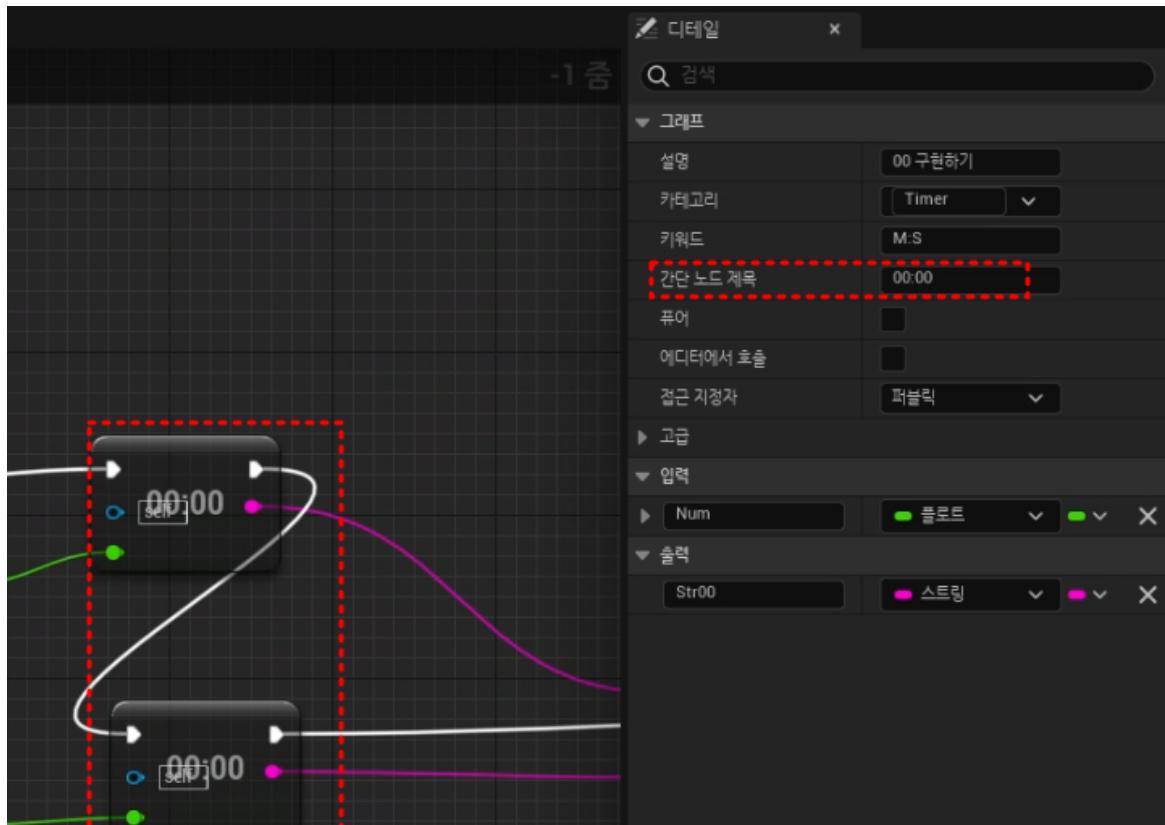
## 키워드

- 액션을 검색할 때 내가 설정한 키워드를 입력하면 해당 함수가 나오는 것을 볼 수 있다



## 간단 노드 제목

- 이벤트 그래프 창에서 내가 입력한 제목이 노드 가운데 나타나는 것을 확인할 수 있다



## 퓨어

- 노드를 간소화할 수 있다

\*\* 실행라인이 사라지기 때문에 실행 순서를 알 수 없게 되는 문제가 발생한다

\*\* Get으로 사용할 경우 가능하지만 Set으로 사용할 예정이라면 이 작업을 하면 안된다

