[유니티 기초] 10. 애니메이터 다루기

유니티·개발

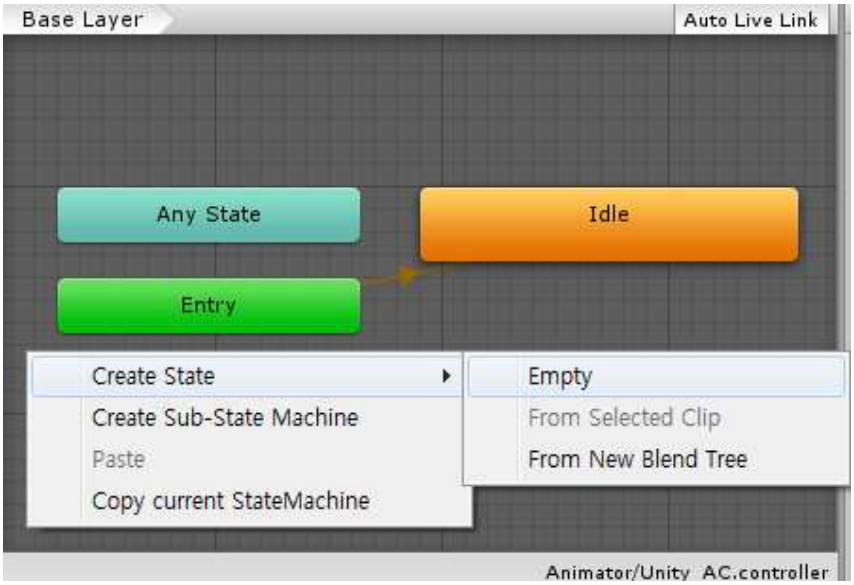
2015. 9. 25. 0:17

http://blog.naver.com/gold_metal/220491375543

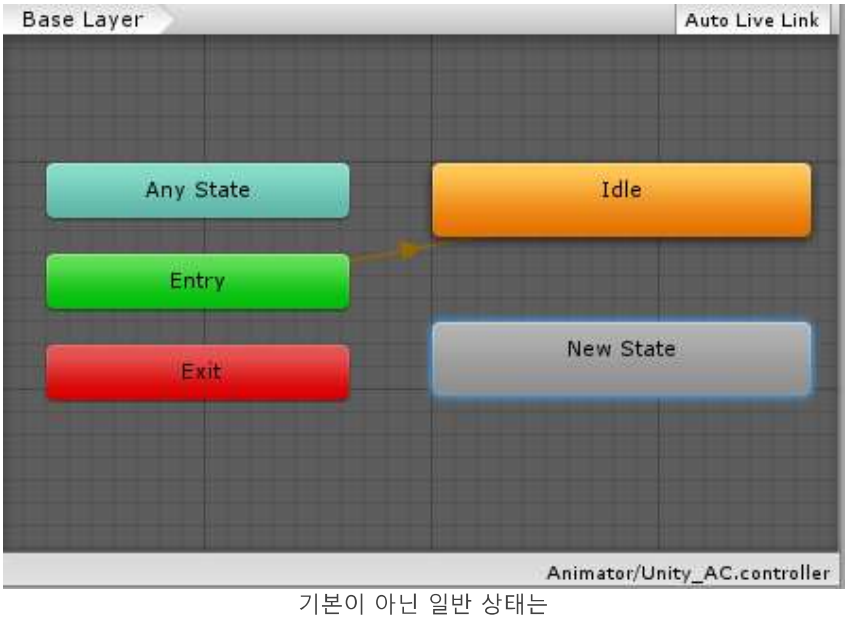
안녕하세요.
어느새 유니티 강좌가 10장까지 나가게 되었군요.

이번에는 메카님을 사용하여
애니메이터를 다루는 전반적인 내용에 대해
알려드리고자 합니다.

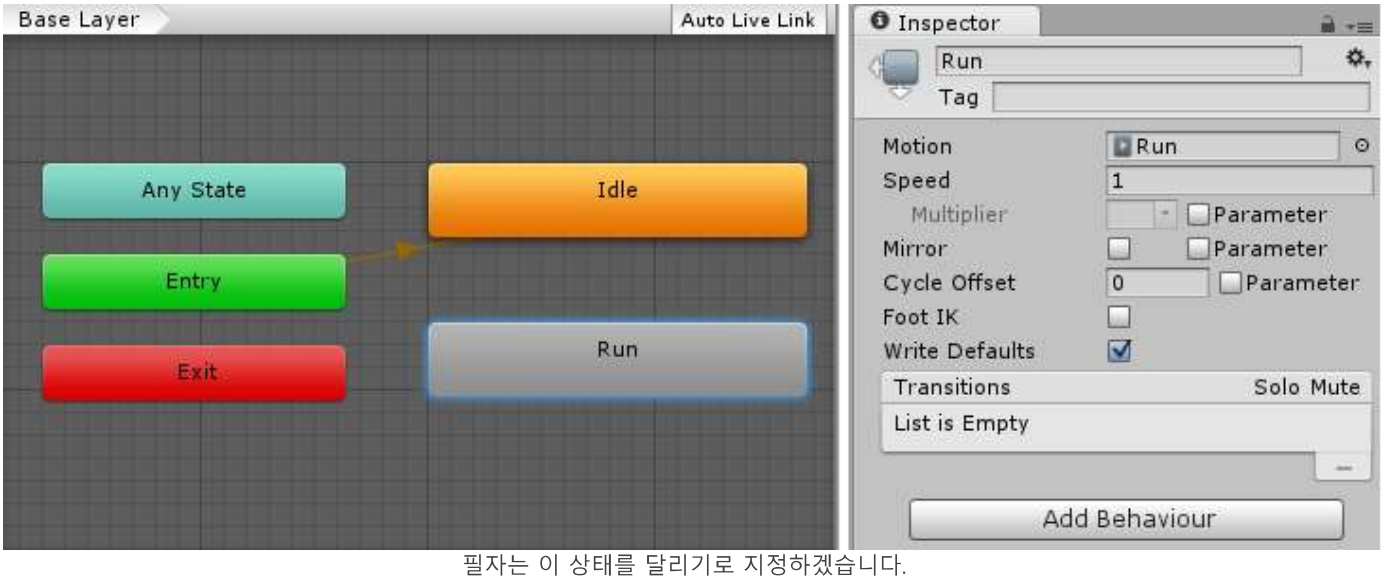
지난 편 내용에 이어서 진행하겠습니다.
기본 애니메이션까지 진도 나갔었지요?



오른쪽 마우스 클릭으로
새로운 상태를 만들어줍니다.

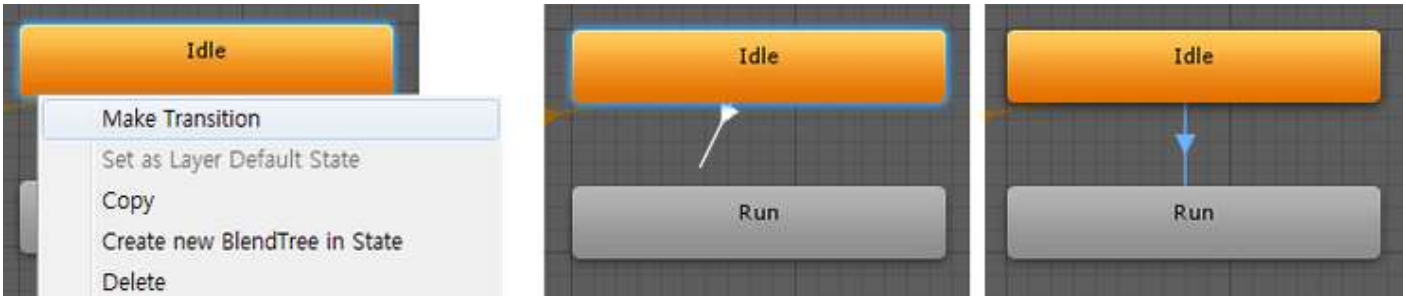


기본이 아닌 일반 상태는
이렇게 회색 네모로 나타납니다.



필자는 이 상태를 달리기로 지정하겠습니다.
달리기 모션을 넣어주고 이름도 지어주었습니다.

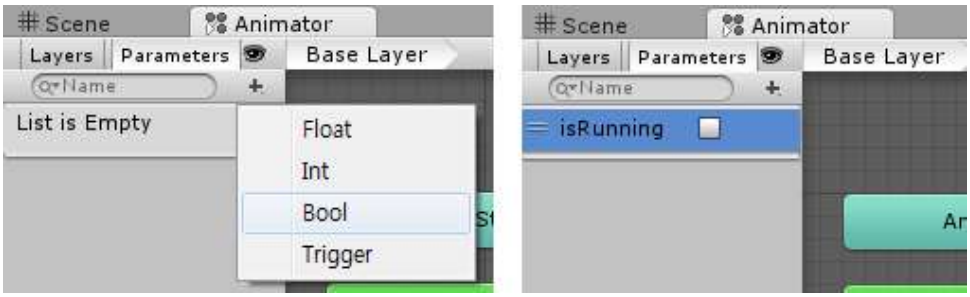
두 상태는 상황에 따라 전환되야 하므로
서로 연결시켜주겠습니다.



상태 하나를 선택하고 오른쪽 마우스 클릭 메뉴에서
Make Transition 으로 전환 시점을 만들어 준 다음,
연결해야 할 상태를 클릭하면 연결 완료입니다.

이제 전환할 타이밍을 만들어야 합니다.
흔히 말하는 플래그(Flag) 라고도 하죠.

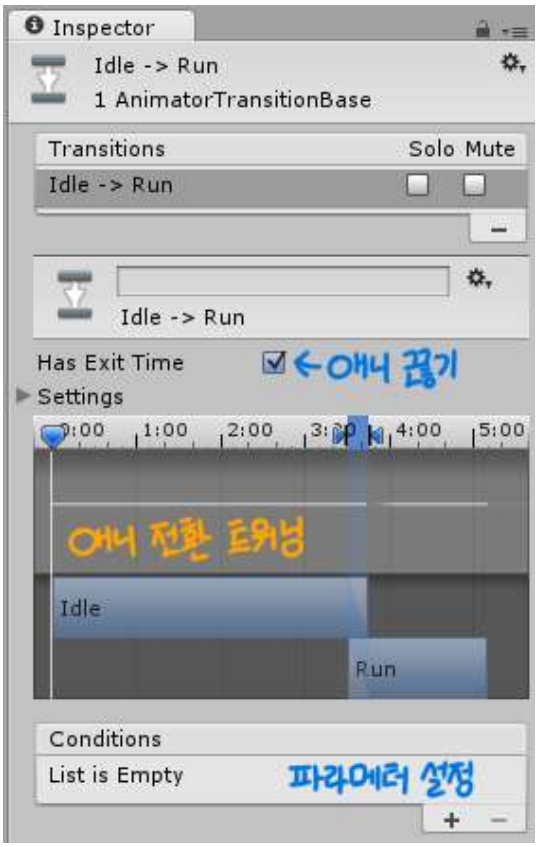
유니티에서는 매개변수, **파라메터(Parameter)**라고 부릅니다.



애니메이션 창의 왼쪽에는
각종 파라메터를 만들 수 있는 곳이 있습니다.

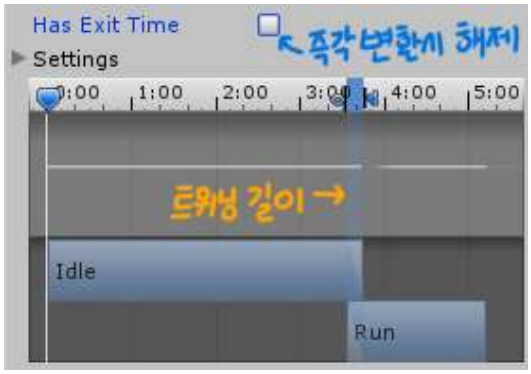
필자는 **움직이는가?** **움직이지 않는가?**
이 두가지 상황으로 상태를 전환해보겠습니다.
그럴려면 **Bool** 타입이 제격이죠.

타입을 선택한 후, 이름을 기입해줍니다.



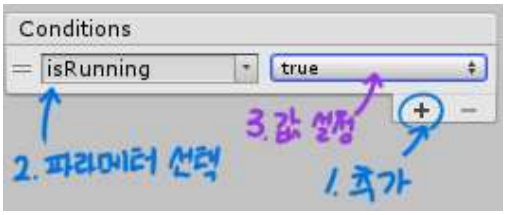
이제 오른쪽의 전환시점 속성 창에 대해 살펴보도록 합시다.

Has Exit Time 체크박스와
애니메이션 길이 같은 것도 보이네요.

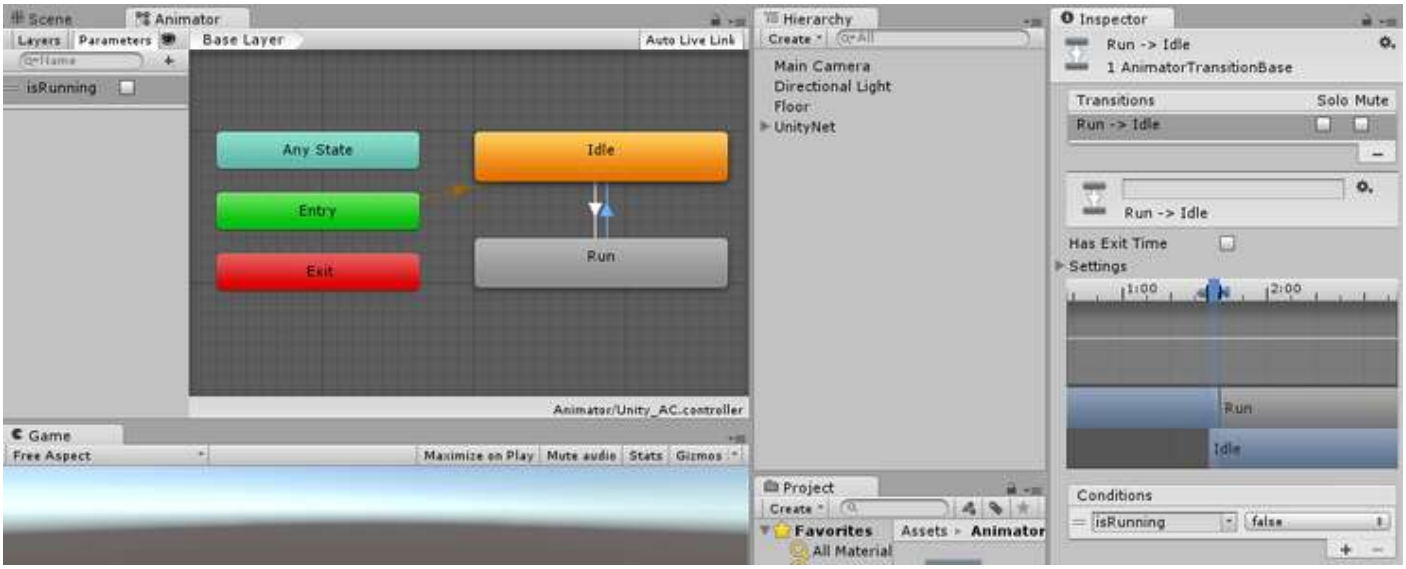


Has Exit Time
전환 시점이 바뀌어 다른 상태가 되어도
현재 재생 중인 애니메이션을 끝날 때까지
강제로 유지하는 옵션입니다.

Settings
두 애니메이션이 부드럽게 전환되도록
트위닝 길이를 설정하는 곳입니다.
겹치는 구간이 길수록 길고 느리게 전환됩니다.

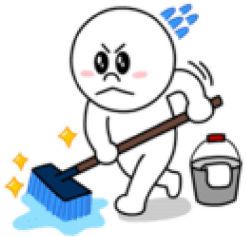


Conditions
상태가 전환될 조건을 설정합니다.
조건은 왼쪽에서 만들어둔 파라메터를 사용하며,
여러 개를 중첩하여 설정할 수 있습니다.



이런 방법으로 각각의 상태를
의도한 전환 시점대로 연결해주면 됩니다.

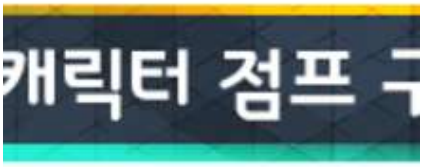
이렇게 설정이 끝난 후엔?
스크립트에서 사용해주어야겠지요.



스크립트 작성은 힘든 작업이지요.
하나씩 하나씩 코드를 눈에 익히면서

차근차근 실력을 갈고닦아 나가면
어려울 것이 없답니다.

스크립트는 점프까지 구현했던
6편에 이어서 진행합니다.



[유니티 기초] 6. 캐릭터 점프 구현
안녕하세요.골드메탈입니다.이번 편에서는 스크립트로점프를 구현하는 방법에 대해 다루어보겠습니다.※ 해당 강좌는 프로그래밍에 대한 기초 지식...
blog.naver.com

```
Rigidbody rigdbody;
Animator animator;  ← 1. 선언

Vector3 movement;
float horizontalMove;
float verticalMove;
bool isJumping;

//-------------------------------------------------------[ Override Function ]

void Awake ()
{
    rigdbody = GetComponent<Rigidbody> ();
    animator = GetComponent<Animator> ();  ← 2. 불러오기
}
```

리지드바디를 불러왔던 것처럼
애니메이터도 가져옵니다.

```
void Update ()
{
    horizontalMove = Input.GetAxisRaw ("Horizontal");
    verticalMove = Input.GetAxisRaw ("Vertical");

    if(Input.GetButtonDown("Jump"))
        isJumping = true;

    AnimationUpdate();
}

void FixedUpdate ()
{
    Run ();
    Jump ();
}

//-----------------------------------------------------[ Animation Function ]

void AnimationUpdate ()
{

}
```

애니메이션만을 처리하기 위한
함수를 만들어 주었습니다.

```
void AnimationUpdate ()
{
    animator.SetBool("isRunning",false);
}
```

이제 애니메이터의 파라메터를 지정해보죠.
필자가 만들었던 'isRunning' 파라메터는 Bool 타입이었습니다.
때문에 **SetBool** 함수를 사용해야 하죠.

이처럼 파라메터의 타입마다
함수도 따로 준비되어 있다는 사실.

```
void AnimationUpdate ()
{
    if(horizontalMove == 0 && verticalMove == 0){
        animator.SetBool("isRunning",false);
    }
    else {
        animator.SetBool("isRunning",true);
    }
}
```

이런 방법으로 파라메터를 상황에 맞도록 지정합니다.

필자는 이동 키 입력이 없으면 대기상태,
아니면 달리기 상태로 지정하였습니다.

필자가 의도한대로 애니메이션이 잘 전환되는 모습입니다.

애니메이션 창을 열어놓은 상태로 게임을 실행하면
이처럼 메카님 안에서 진행 상황을 실시간으로 확인할 수 있습니다.

열번째 강좌 잘 보셨는지요.
다음 시간은 여태까지 배웠던 내용을 토대로
캐릭터의 기본 행동 구현에 대해 다루어보겠습니다.

그럼.