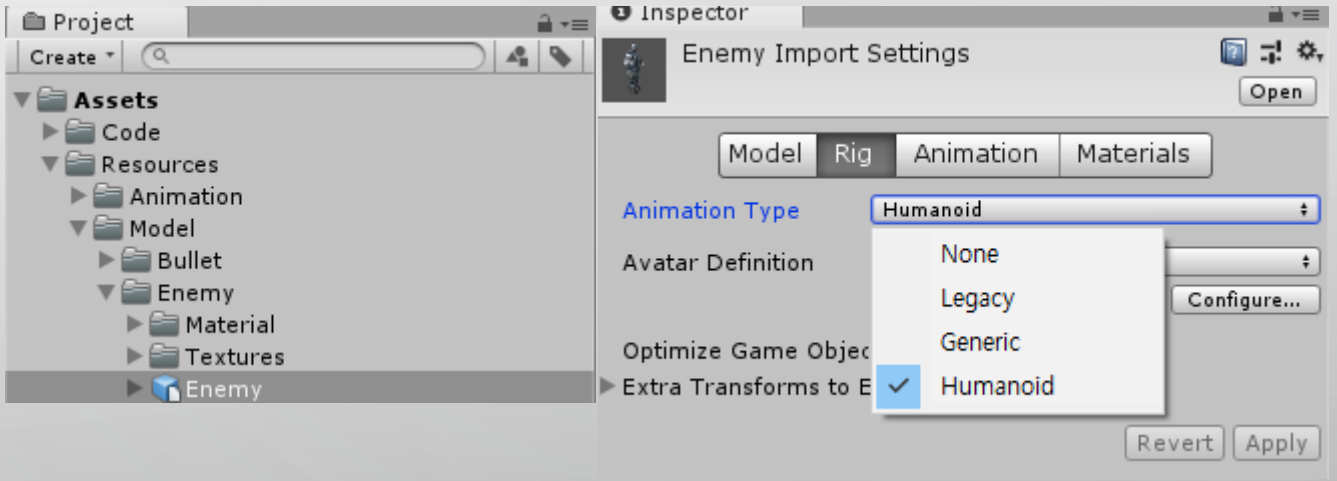
UNITY 3D
-CHAPTER7-

SOULSEEK

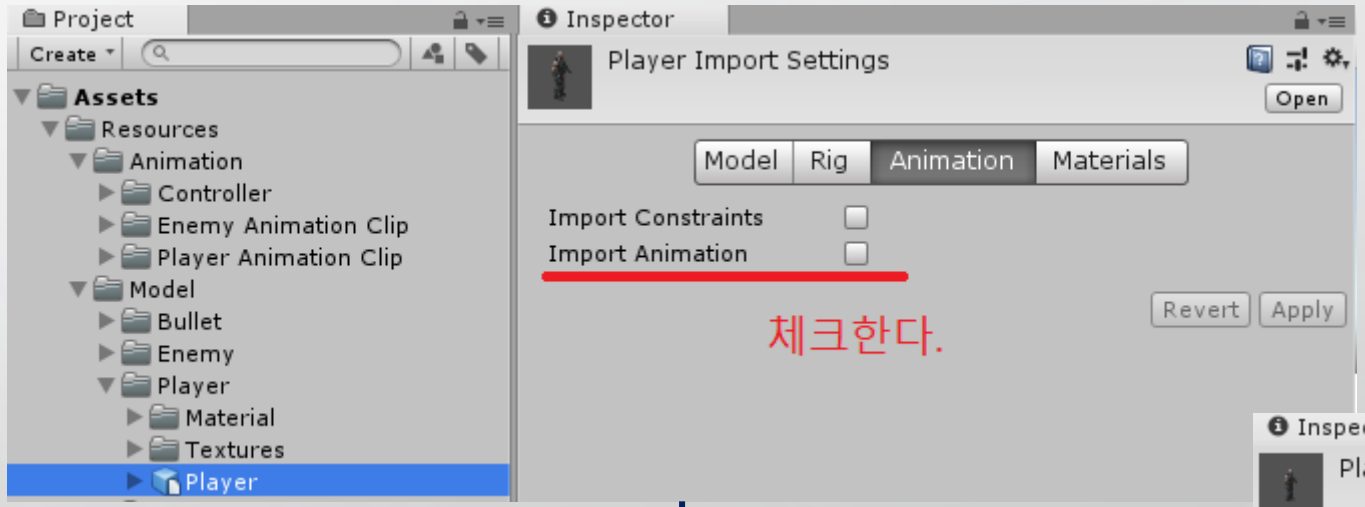
UNITY의 ANIMATION

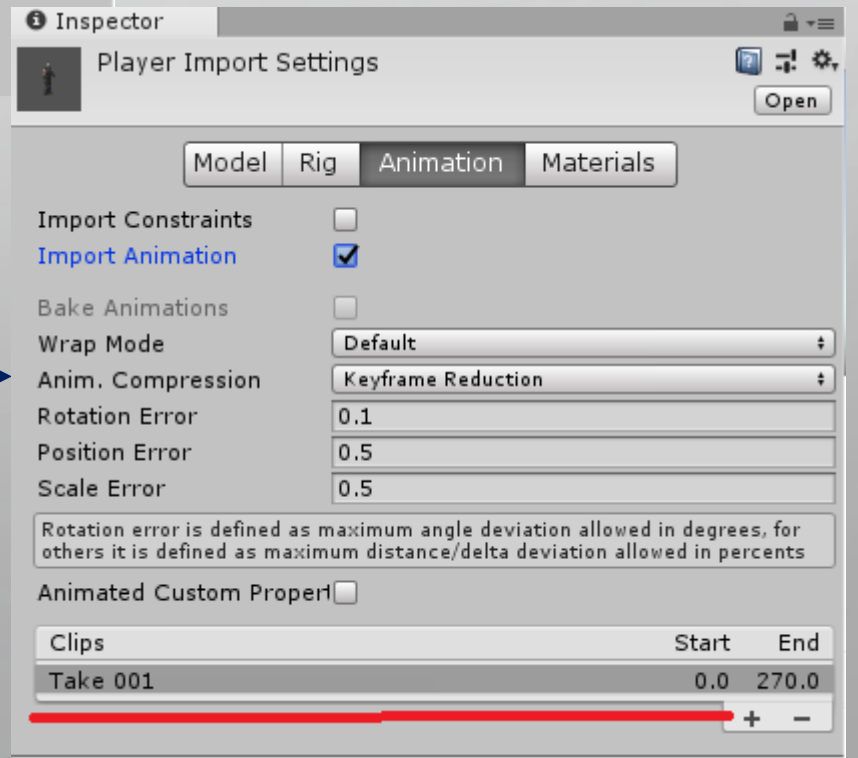
- 레거시 : 하위 호환성을 위해 사용(4.x버전)
- 제네릭 : 메카님 애니메이션, 비인간형, 리타겟팅 기능 사용 불가.
- 휴머노이드 : 메카님 애니메이션, 2족보행 혹은 인간형, 리타겟팅 가능.
- 메카님 애니메이션은 Animator을 사용해서 Animation 컨트롤을 한다.



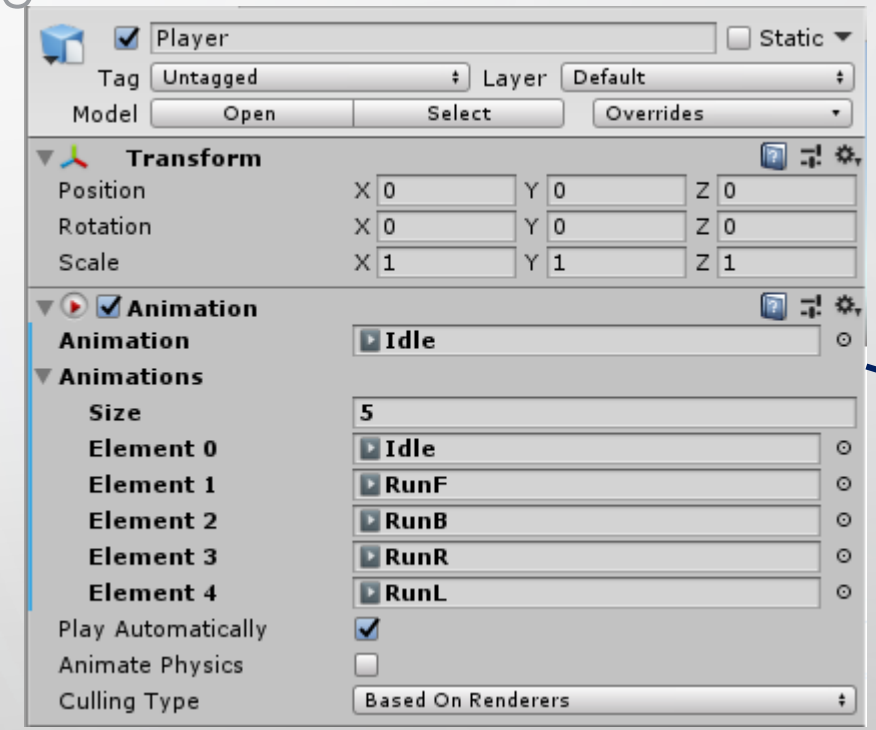
LEGACY ANIMATION

- 속도면에서 빠르기 때문에 Legacy를 아직 많이 쓰지만 하위 호환만 시켜줄 뿐 더 이상의 지원이 없기 때문에 Mecanim을 쓰는 것이 좋다.





LEGACY ANIMATION



기본 동작 애니메이션과 사용할 애니메이션 클립을 만들어 준다.

LEGACY ANIMATION

```
[System.Serializable]                    ①
public class PlayerAnim
{
    public AnimationClip idle;
    public AnimationClip runF;
    public AnimationClip runB;           ②
    public AnimationClip runR;
    public AnimationClip runL;
}

public enum STATE
{
    PLAYER_IDLE = 0,
    PLAYER_MOVE,
    PLAYER_FIRE,                         ③
    PLAYER_DAMAGE,
    PLAYER_WIN,
    PLAYER_DIE = 99,
}
```

1. Class는 publi으로 선언하여도 insfecr view에 노출이 되지 않는다. 이와 같이 선언하면 노출 할 수 있다.
2. Animation클립을 추가 한 Animation Comonent의 것과 같은 클립들을 추가해 준다.
3. Player 상태를 체크해서 플레이 시켜준다.

```
void Start()
{
    trans = GetComponent<Transform>();
    anim = GetComponent<Animation>();
    anim.clip = playerAnim.idle;
    anim.Play();
}
```

- 기본 상태의 애니메이션 클립을 설정해주고 플레이 시켜준다.
- 이 후 상태에 따라 애니메이션 클립만 바꿔주면 된다.

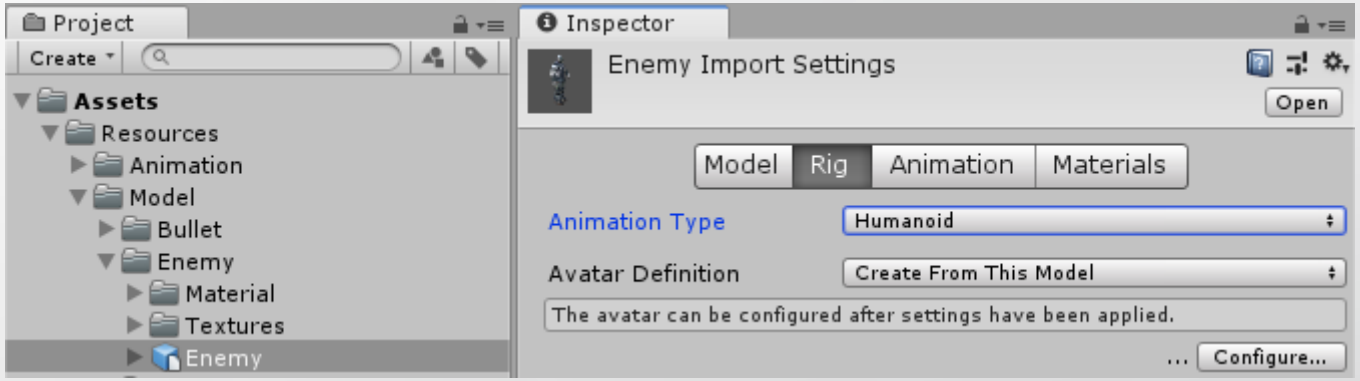
LEGACY ANIMATION

```
void AnimationStart(float ho, float ve)
{
    if (ve >= 0.1f)
    {
        anim.CrossFade(playerAnim.runF.name, 0.3f);
    }
    else if (ve <= -0.1f)
    {
        anim.CrossFade(playerAnim.runB.name, 0.3f);
    }
    else if (ho >= 0.1f)
    {
        anim.CrossFade(playerAnim.runR.name, 0.3f);
    }
    else if (ho <= -0.1f)
    {
        anim.CrossFade(playerAnim.runL.name, 0.3f);
    }
    else
    {
        anim.CrossFade(playerAnim.idle.name, 0.3f);
    }
}
void Operate()
{
    horizontal = Input.GetAxis("Horizontal");
    vertical = Input.GetAxis("Vertical");

    AnimationStart(horizontal, vertical);

    Vector3 vDir = (Vector3.forward * vertical) + (Vector3.right * horizontal);

    trans.Translate(vDir.normalized * moveSpeed * Time.deltaTime, Space.Self);

    rot = Input.GetAxis("Mouse X");

    trans.Rotate(Vector3.up * rotSpeed * rot);
}
```

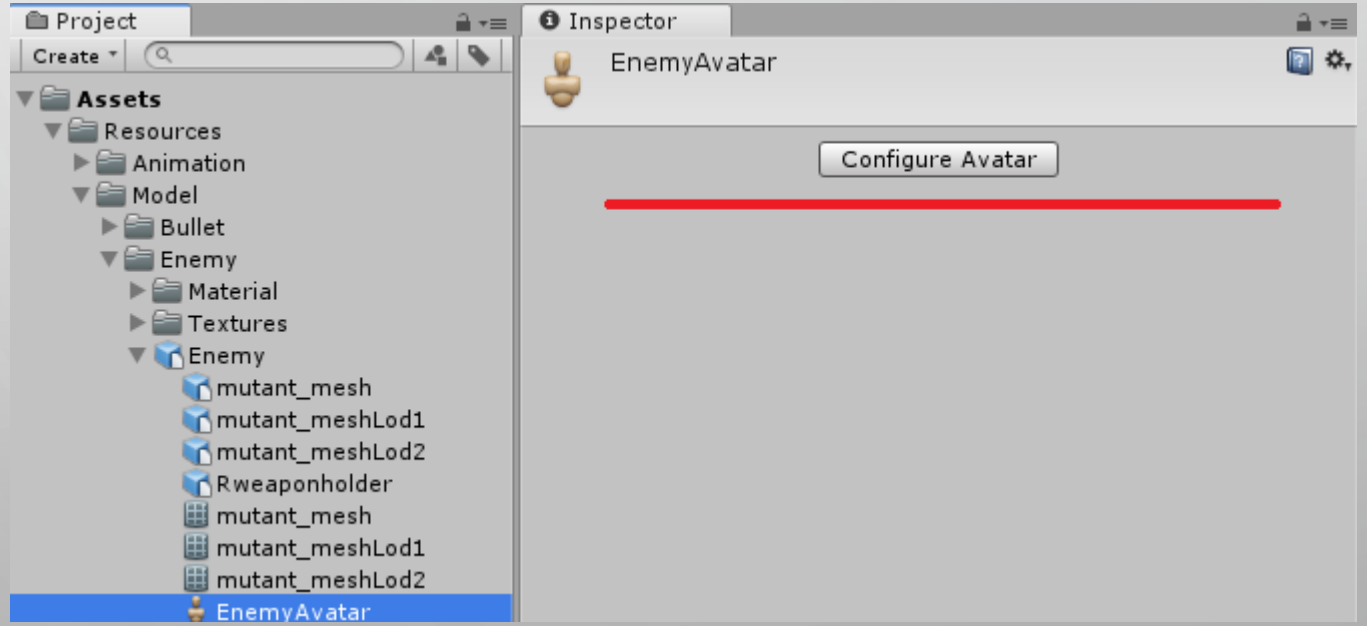
- Animation 을 크로스 페이드로 자연스럽게 변화 시켜줄 클립으로 전환 시켜준다.

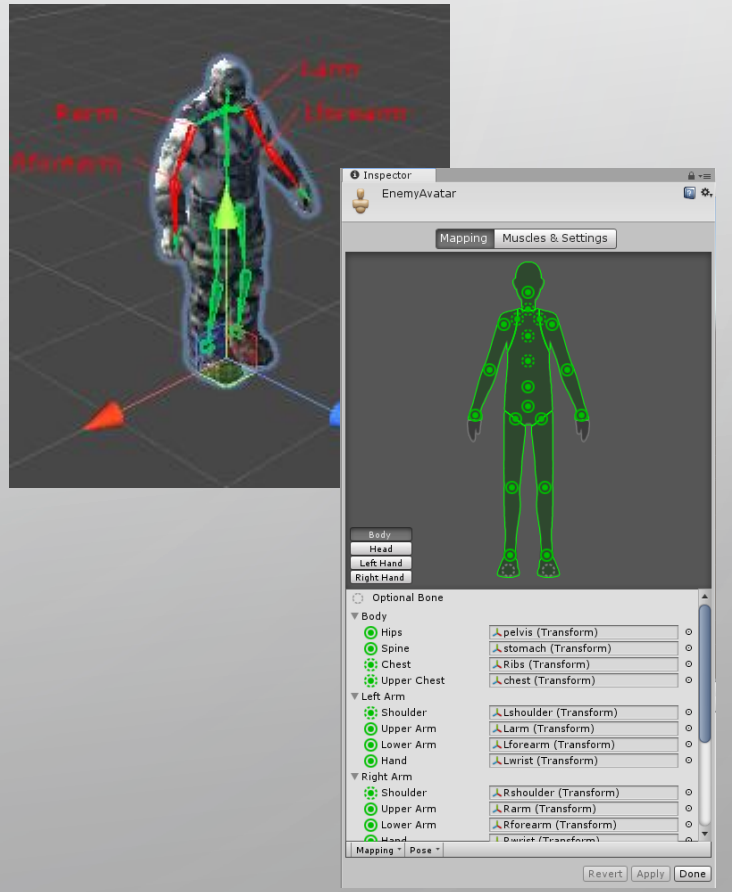
- InputManagerView에서 설정된 이동처리 값을 읽어 온다.
- 전후 좌우 처리 값을 알아 온 뒤 노멀 벡터로 방향 값만 체크한다.
- 정해둔 스피드를 가지고 자기좌표를 중심으로 움직여 준다.
- 마우스 위치로 좌우 회전값을 처리한다.

MECANIM ANIMATION

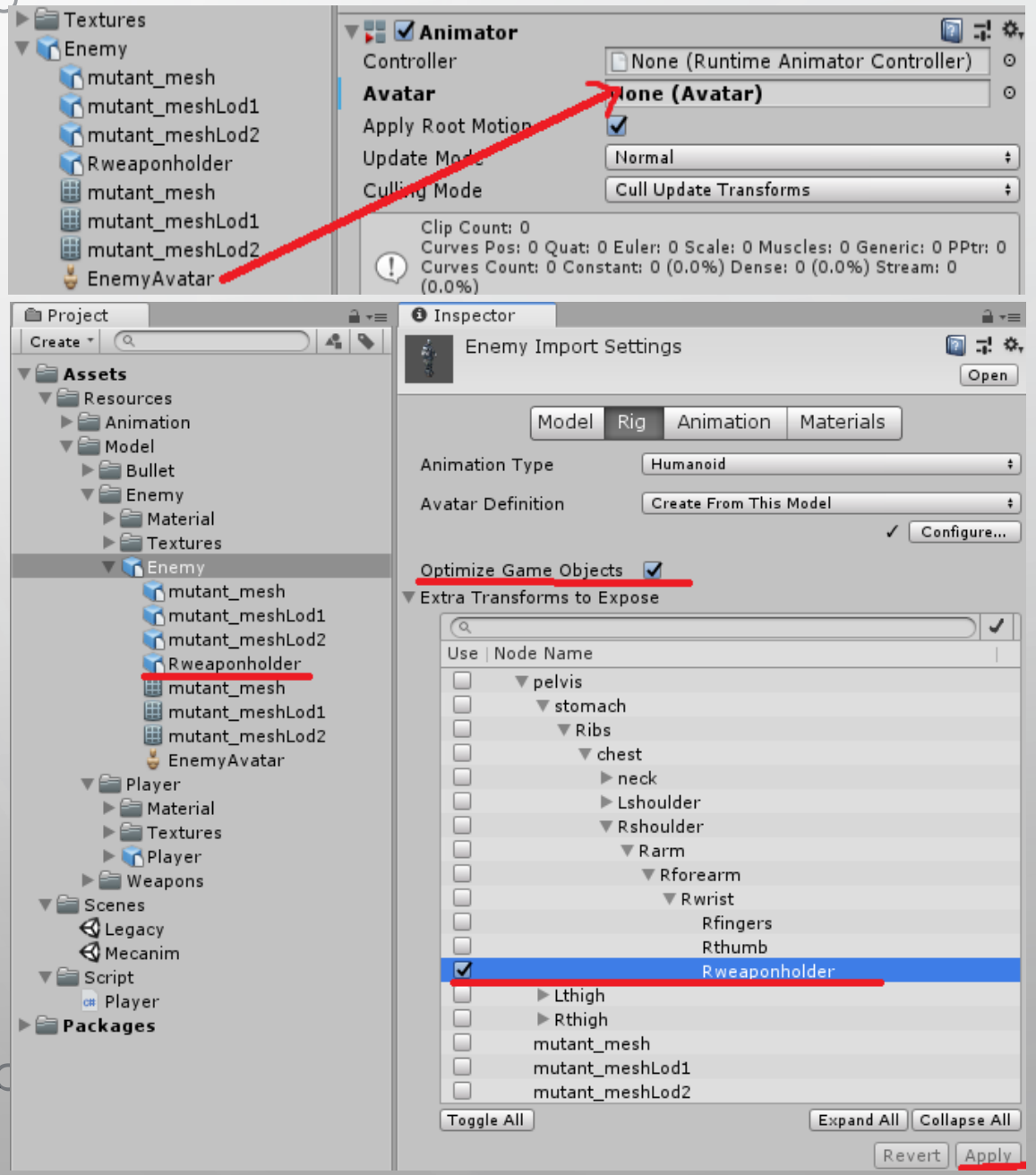


모델링을 클릭해서 rig을 선택해서 휴머노이드로 바꿔주자 아래에 보면 아비터라는 것이 있는데 그것을 눌러서 보면 IK정보등 여러가지를 볼 수 있고 나중에 IK를 조작해서 동작이나 비활성화등 다양한 동작을 할 수 있다.





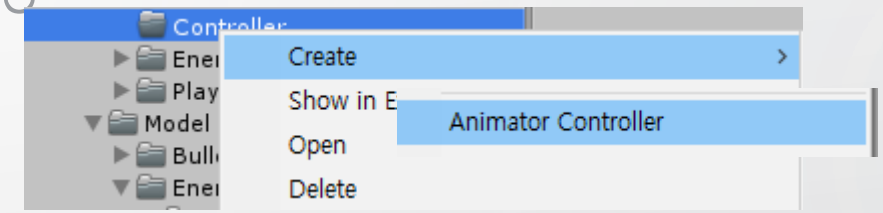
MECANIM ANIMATION



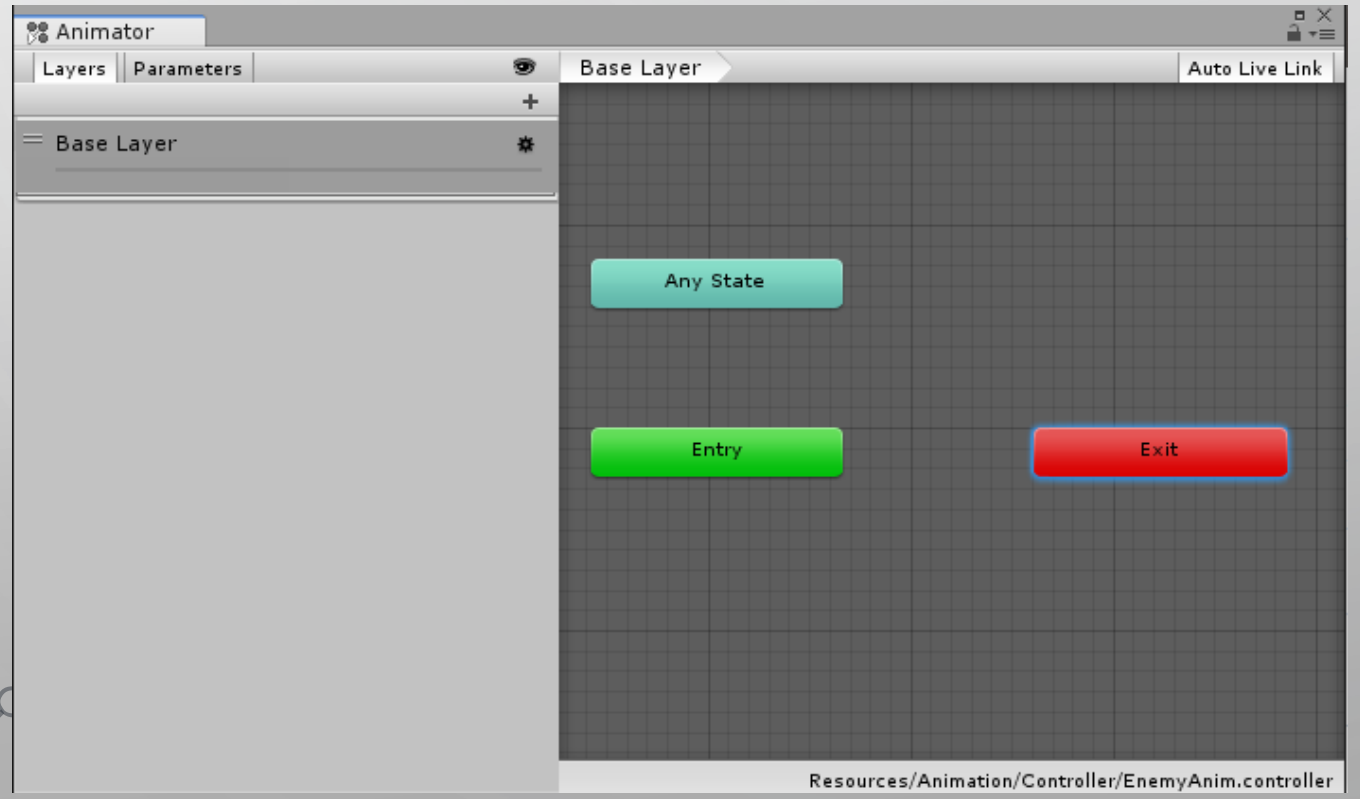
- Animator에 준비된 Avatar연결

- Optimize Game Object를 클릭한다.
- 하위 객체 중 보여줄 것을 선택하는 것이다.
- 예를 들어 총기를 장착할 포지션을 보여주고 싶을 때 해당할 곳을 보여주고 체크 한 뒤 Apply한다.

MECANIM ANIMATION

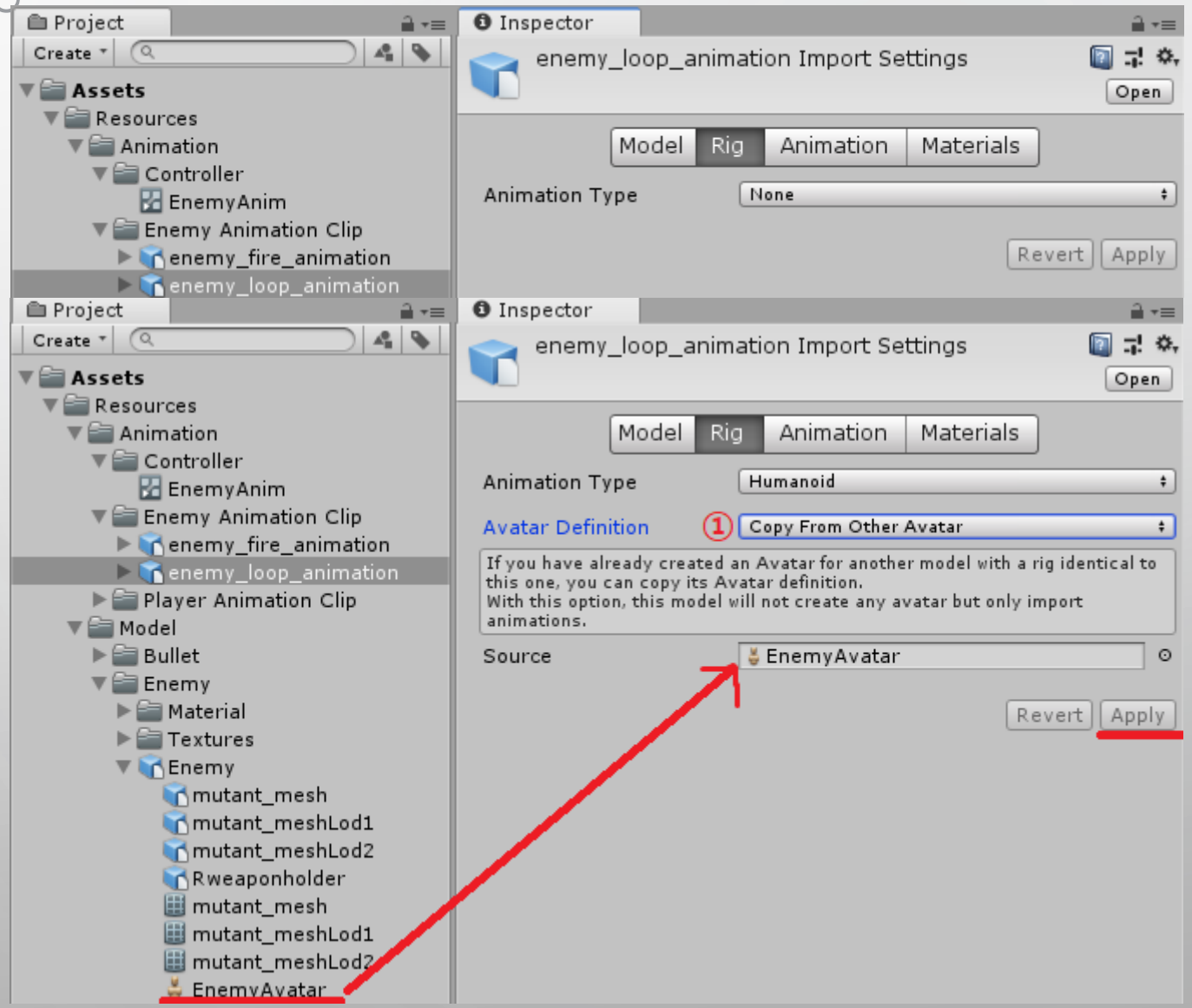


- 이제 Animator를 만들어 보자.

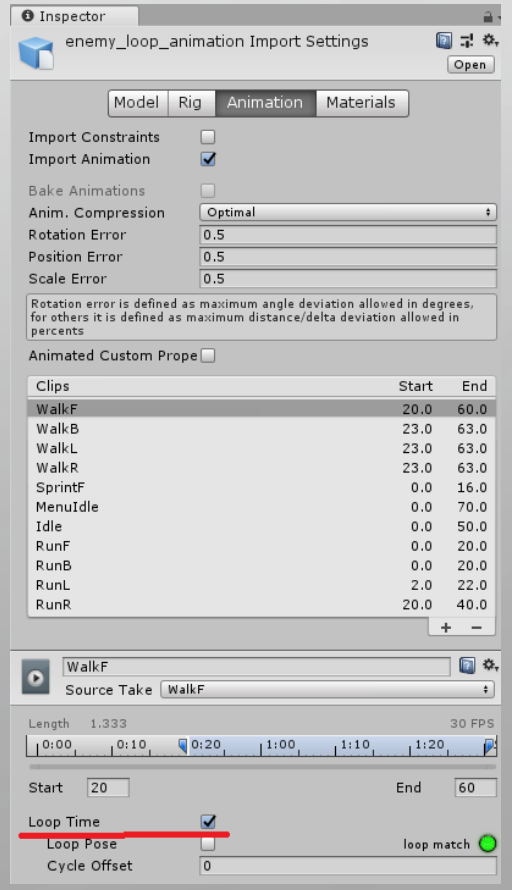


- Entry : 시작 스테이트로 최조 상태랑 연결 된다.
- Any State : 조건 만족 시 바로 변경해야 할 상태가 있다면 적용
- Exit : 종료 스테이트

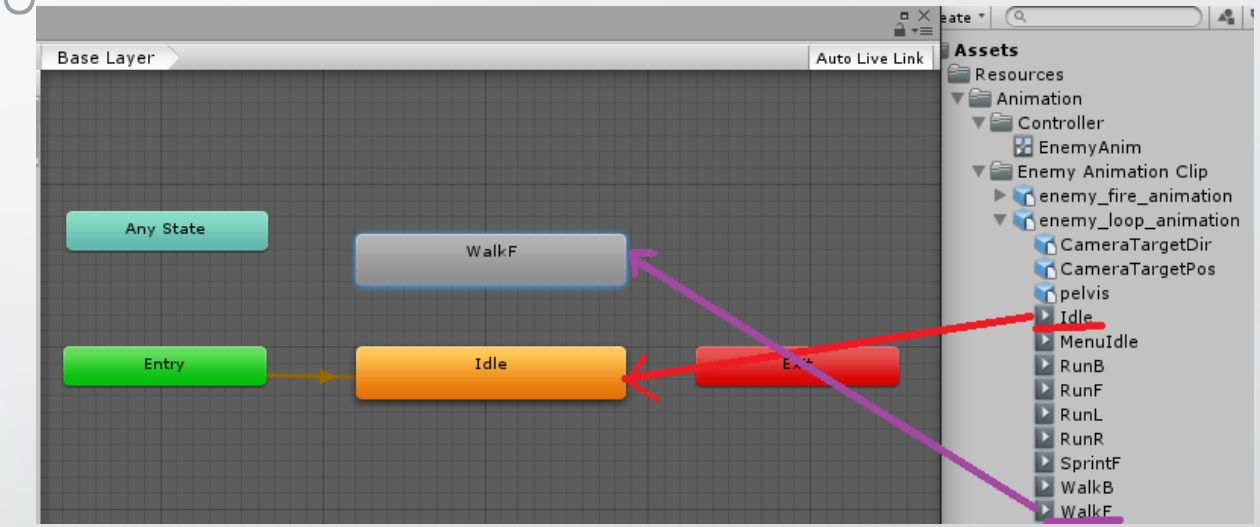
MECANIM ANIMATION



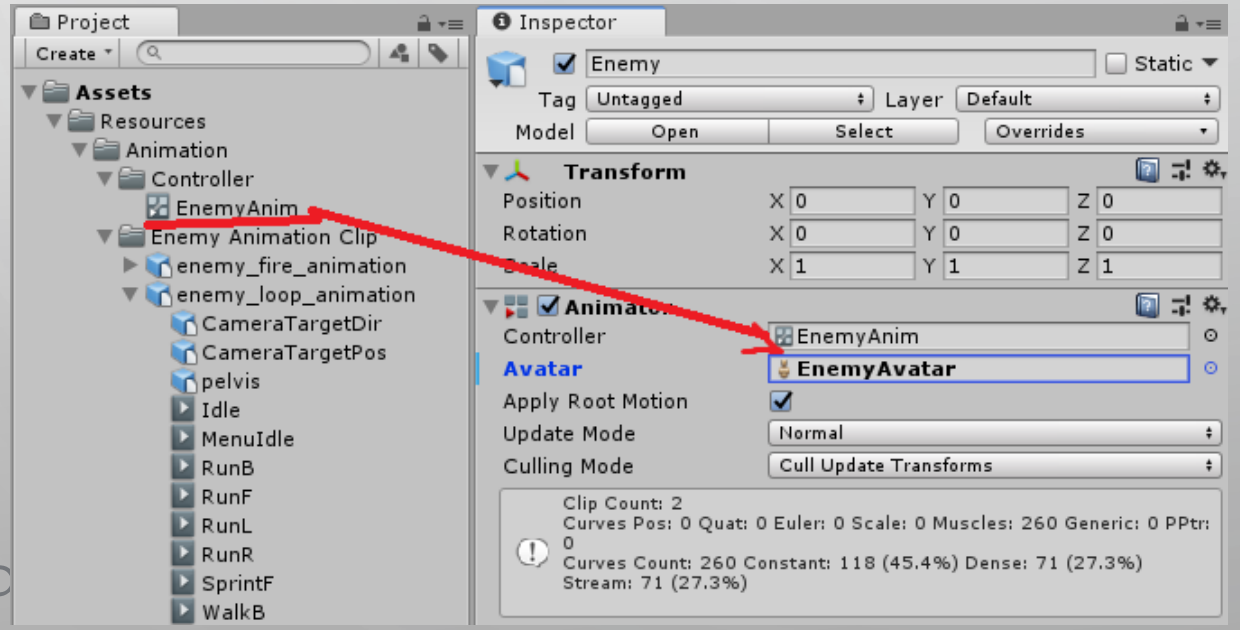
- 준비된 Animation을 설정하자.
- 모델링을 humanoid를 했으니까 똑같이 설정하자.
- Avatar를 카피하는 것으로 바꿔주고 준비된 Avatar를 연결 시켜주자.
- Apply해준다.



MECANIM ANIMATION

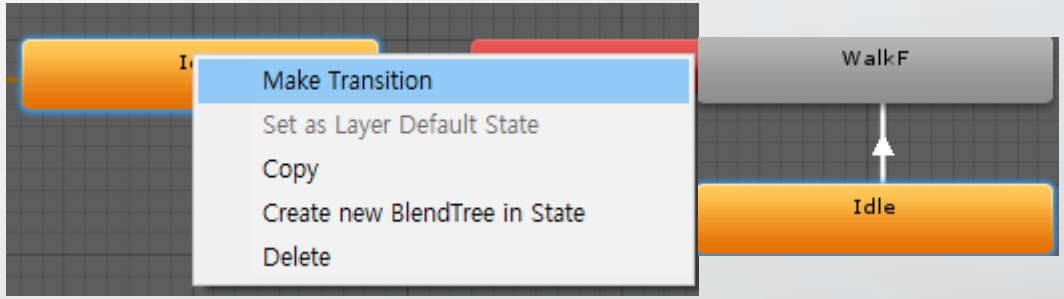


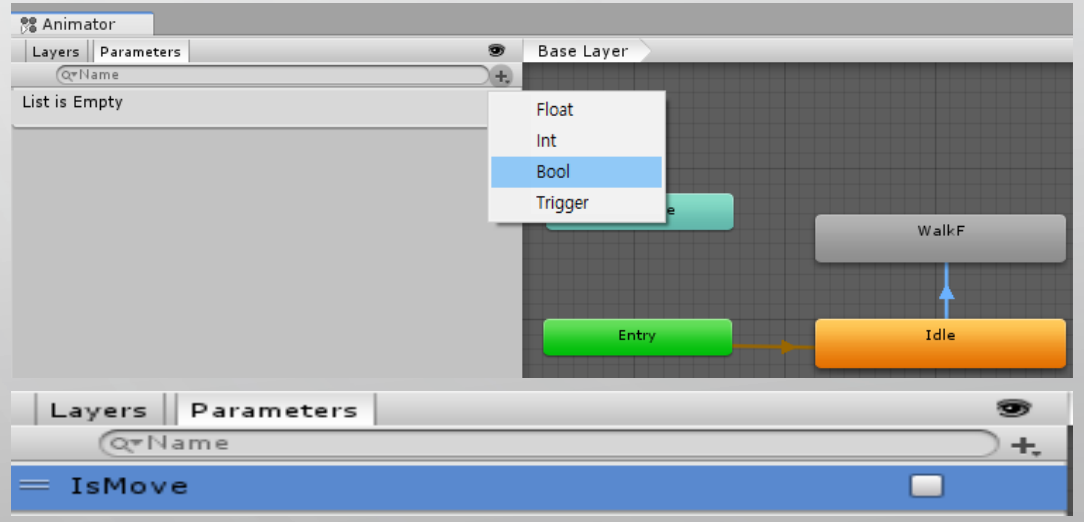
- 준비된 Animator창에 필요한 Animation Clip을 추가하자
- 제일 처음 추가된 것이 시작으로 인식되어 Entry와 자동으로 연결된다.



- 하이러키 창에 Animator에 생성한 Animator를 추가해서 테스트 해보자.

MECANIM ANIMATION

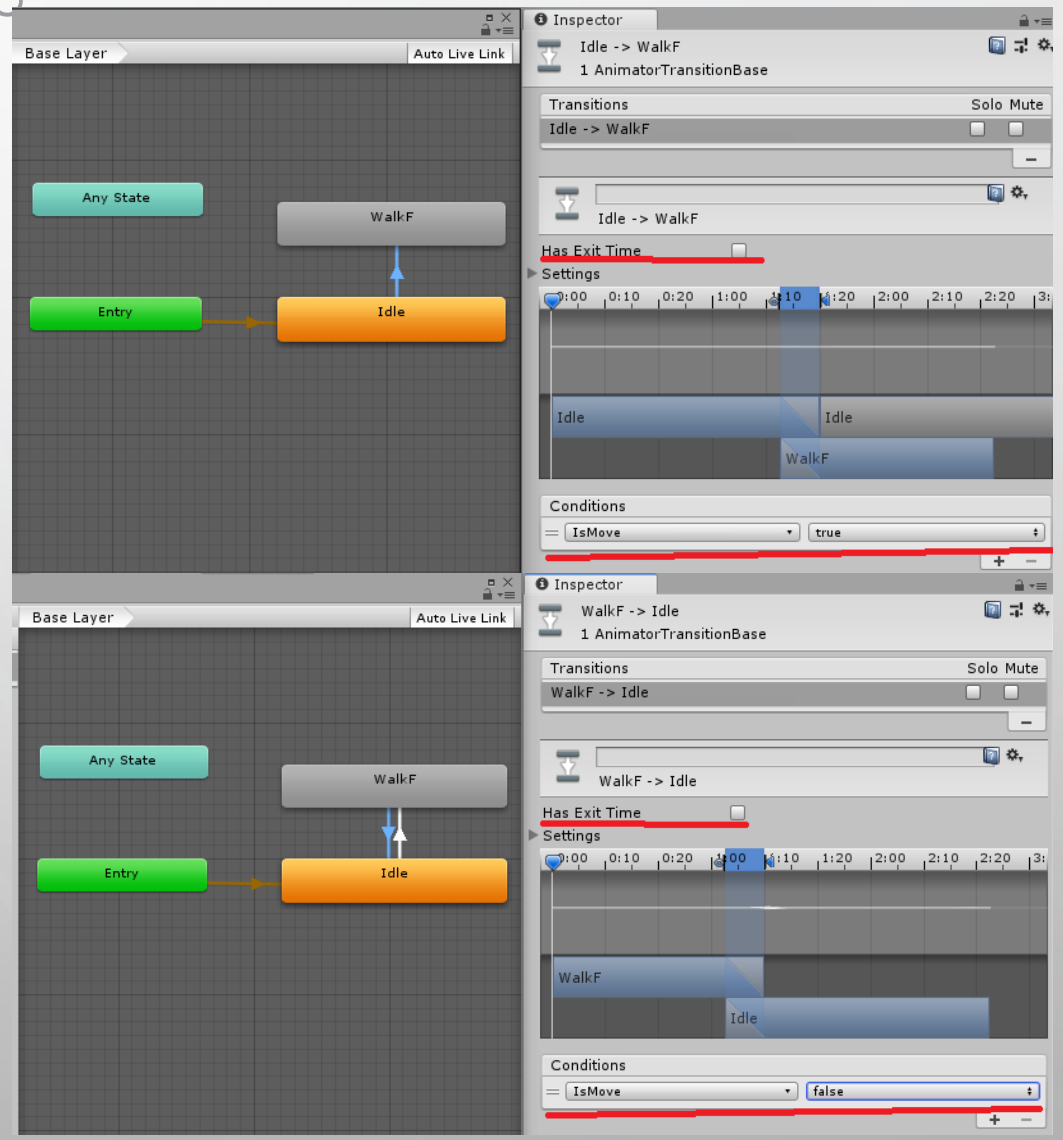




- 추가된 Clip에 우클릭을 하여 Make Transition을 생성해서 연결할 Clip으로 드래그 해주면 Transition이 연결되고 전이를 할 수 있게 된다.

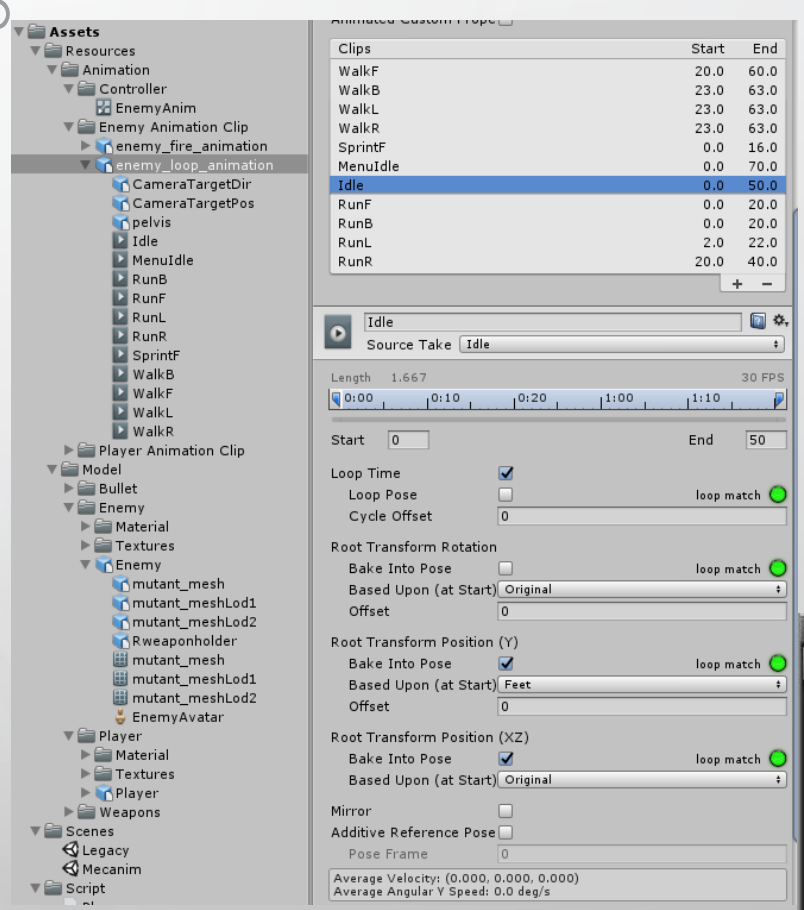
- Paramaters를 클릭해서 Trasition할 Paramater를 추가하자 4종류의 방식으로 사용 할 수 있다.

MECANIM ANIMATION



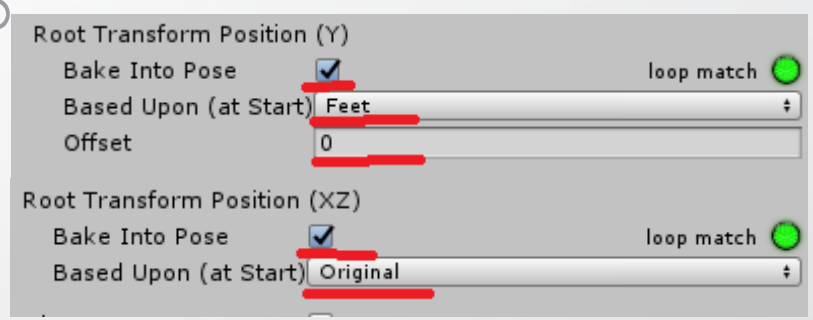
- 서있는 자세와 걷는 자세를 움직일 때 반복하므로 서로 전이되게 만들자.
- 각 Transition을 클릭해보면 설정을 할 수 있는데 서로 전이가 자연스럽게 해주기 위해 상태가 바뀌어도 이전동작을 모두 한 뒤에 전이되는 설정인 Has Exit Time을 언 클릭 해주자.
- 그리고 하단에 파라미터를 생성해둔 걸 추가해서 설정해주자.

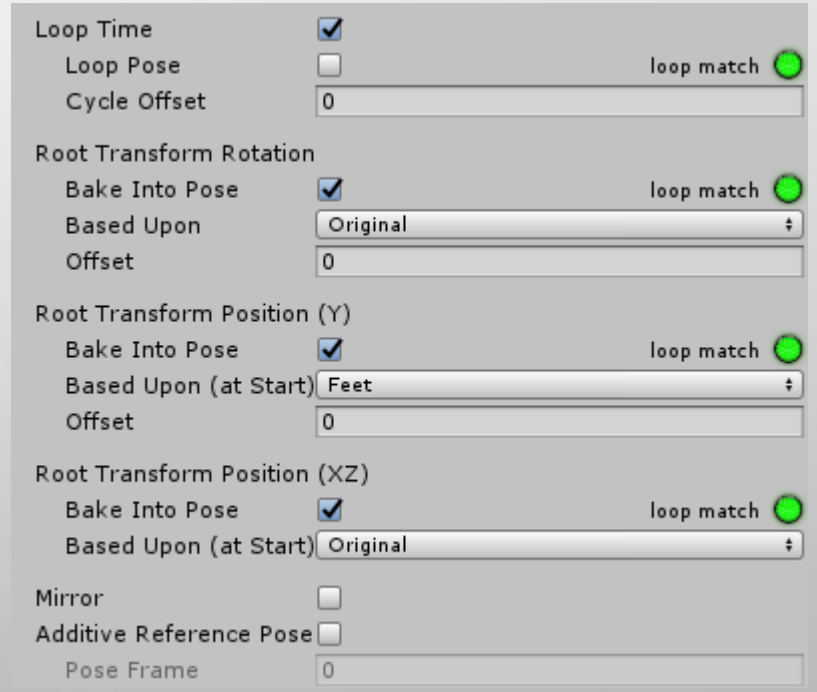
MECANIM ANIMATION



- 만약 해당 애니메이션이 루트 모션 캡쳐로 이루어졌다면 움직일 때 미세하게 좌표가 바뀌게 된다. 그것을 수정할 필요가 있다.

MECANIM ANIMATION





- Idle 상태일때 본의 기준으로는 0의 위치에 있지만 모델링은 그 아래로 내려가 있을 수 있다 그것을 본위치로 옮겨줘야 하기 때문에 Y축을 feet을 해서 축을 본 위치에 맞춰준다.
- XZ 값도 해당 옵션으로 고정해서 바꿔준다.
- WalkF일 때도 바꿔준다.
- 항상 이 옵션을 만질 필요는 없다 모션캡쳐가 아니라면 거의 쓸 필요가 없을것이다.

MECANIM ANIMATION

```
private bool isMove = false;
private Animator anmator;
private readonly int hashMove = Animator.StringToHash("IsMove");

// Start is called before the first frame update
void Start()
{
    anmator = GetComponent<Animator>();
}
if(isMove)
{
    anmator.SetBool(hashMove, true);
    transform.Translate(transform.forward * 0.01f, Space.World);
}
else
{
    anmator.SetBool(hashMove, false);
}
```

- 보이는 것처럼 추가해둔 상태를 전환해서 애니메이션을 동작하게 하면 해당 상태로 전이하게 된다
- 이것을 응용해서 int형으로 선언해서 여러 상태 값에 따라 변화하는 형태로 전이하게 할 수 있다.
- Any State로 각 상태를 연결해서 해당 상태로 바로 연출하는 것으로 이어 줘도 되고 복합적으로 Any State로 주고 그 안에 세부 상태를 여러 동작을 섞어서 연결동작이 될 때에는 Transition으로 연결하는 방법을 써도 좋다.