**2018180020 박재우(클라이언트) 15주차 기록**

**CBV와 32BitConstants**

저번주에 로드한 모델을 그리려고 했으나 하나만 그려지는 문제가 있었다. CBV를 이용해서 위치 데이터를 루트파라미터에 설정해주고 있었는데 32BitConstants로 하니 잘 되었다. CBV는 상수 데이터를 담을 버퍼를 생성하고 그 버퍼를 쉐이더에 바인딩을 해주는 방식이고 32BitConstants는 데이터 자체를 직접 쉐이더에 바인딩을 해주는 방법이다. 데이터를 직접 넣어주다 보니 누락된 데이터 없이 제대로 들어갔다. 그럼 CBV는 왜 안되었을까 생각해보면 아마도 버퍼를 객체 한 개에 대한 수만큼 만들었기 때문일 것이다. 로드하는 함수를 보면 Mesh를 만들 때 CreateShaderVariables 함수로 버퍼를 생성해 주지만 해당 모델을 사용하는 객체가 총 몇 개가 들어와야 하는지 알 수 없기 때문에 버퍼를 미리 생성할 수 없다. 그래서 객체 한 개만큼의 버퍼만 만들어지고 결국 마지막에 그려지는 객체의 데이터로 업데이트 되면서 한 개만 그려진 것이다.

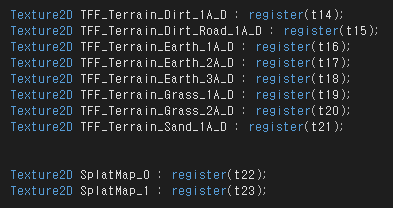
지금 상황으로는 객체가 총 몇 개 그려지는지 알 수 없는 상태이고 넘겨줄 데이터가 float4X4 행렬과 UINT 변수이기 때문에 크기가 크지 않다. CBV는 버퍼를 업데이트 해주는 등의 처리를 하는데 시간이 더 소비되고 데이터는 매 프레임마다 업데이트를 해줘야 한다. 이러한 점을 봤을 때 객체의 위치 데이터는 CBV보다는 32BitConstants를 이용하기로 한다.

**터레인**

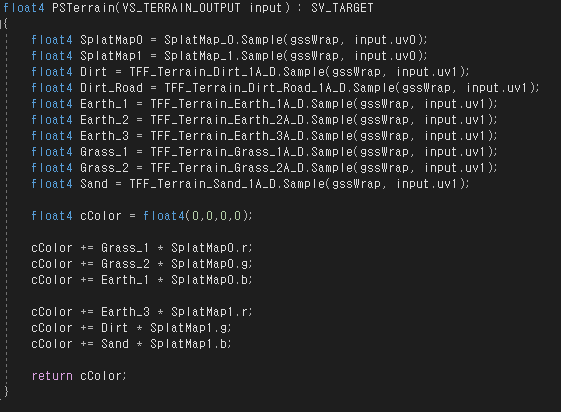
터레인관련 클래스들은 3D 게임 프로그래밍 수업의 과제를 할 때 이용한 클래스들을 이용한다. 과제에서의 터레인은 하나의 Texture와 하나의 Detail Texture로만 만들어졌었다. 때로는 전체의 터레인의 생김새를 나타내는 Texture를 이용해 그려주기도 했다.

우리의 터레인은 Splat Map이라는 것을 사용한다. Splat Map은 터레인에 그려지는 Texture의 강도를 나타내는 이미지로 R G B 채널 각각 담당하는 Texture가 정해져 있고 그 Texture의 강도에 따라 픽셀을 그려주면 된다.

먼저 쉐이더를 만들어준다.



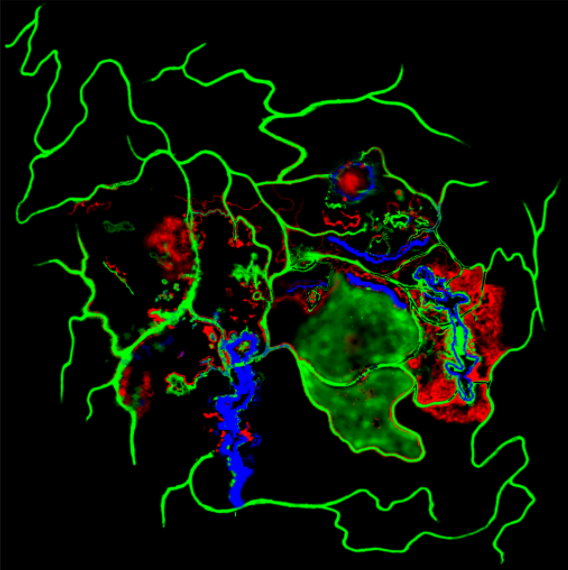
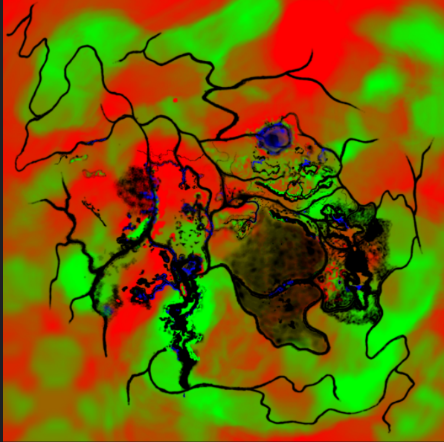
사용할 Texture는 위와 같다. 총 8개의 다른 질감의 Texture이고 우리가 사용할 터레인의 Splat Map은 2개가 있다. 따라서 이 데이터들을 받을 수 있도록 t14~t23까지 할당을 해준다.



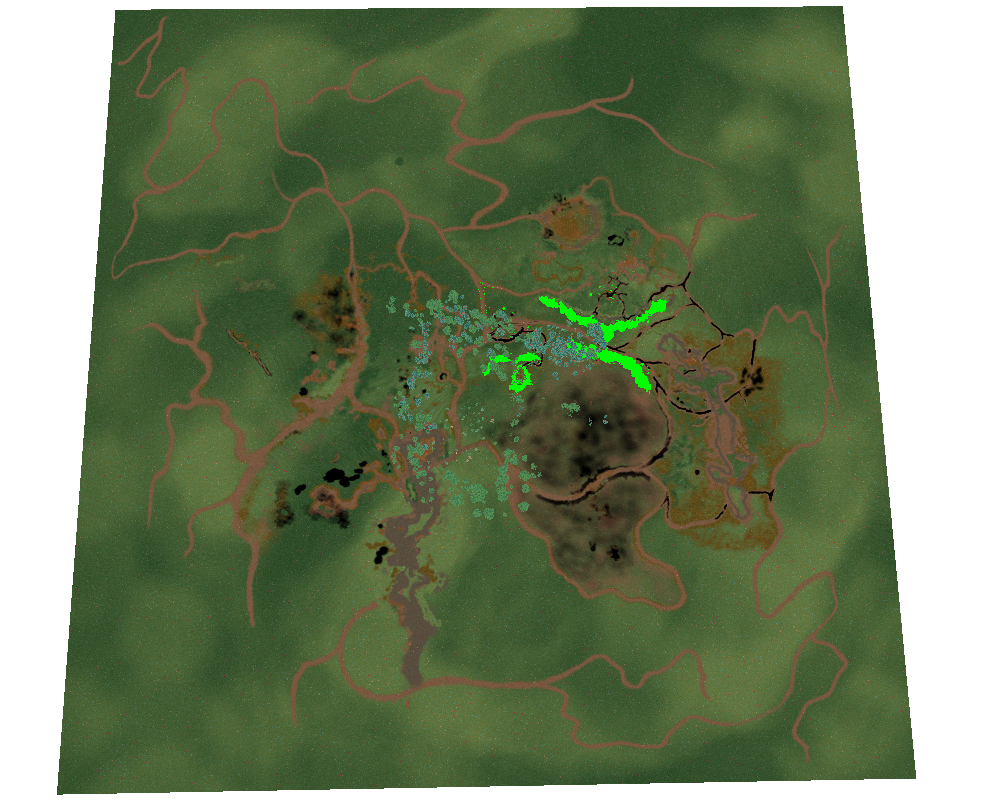
PSTerrain함수는 픽셀 쉐이더이다. 먼저 불러온 Texture의 값들을 float4 변수에 샘플링해서 저장한다. Splat Map은 터레인 전체에 적용되므로 uv0을 사용했고 나머지 Texture들은 하나가 터레인 전체를 덮는 것이 아니라 계속 반복되면서 입혀져야 하기 때문에 uv1을 사용한다.

최종적으로 반환해줄 cColor를 선언해주고 전부 0으로 초기화 해준다.

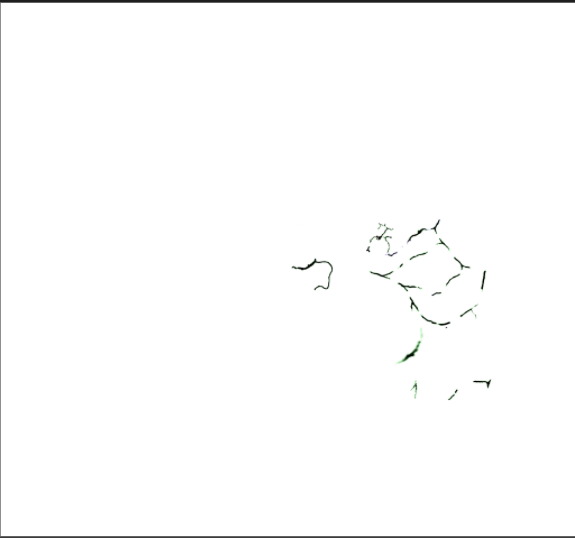
그 후 Splat Map의 R, G, B의 값들을 이용해 cColor에 전부 더해준다. Splat Map 0의 R 채널은 Grass\_1에 해당하는 Texture가 그려지는 강도를 나타낸다. SplatMap0.r 값에 Grass\_1을 곱해서 더해준다. 다른 값들도 똑같이 계산해서 넣어준다.

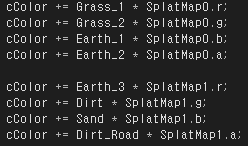


위 두 사진이 Splat Map이다. 각각의 RGB 값은 그려질 Texture의 강도를 의미하고 이들을 이용해 그린 결과물은 다음과 같다.



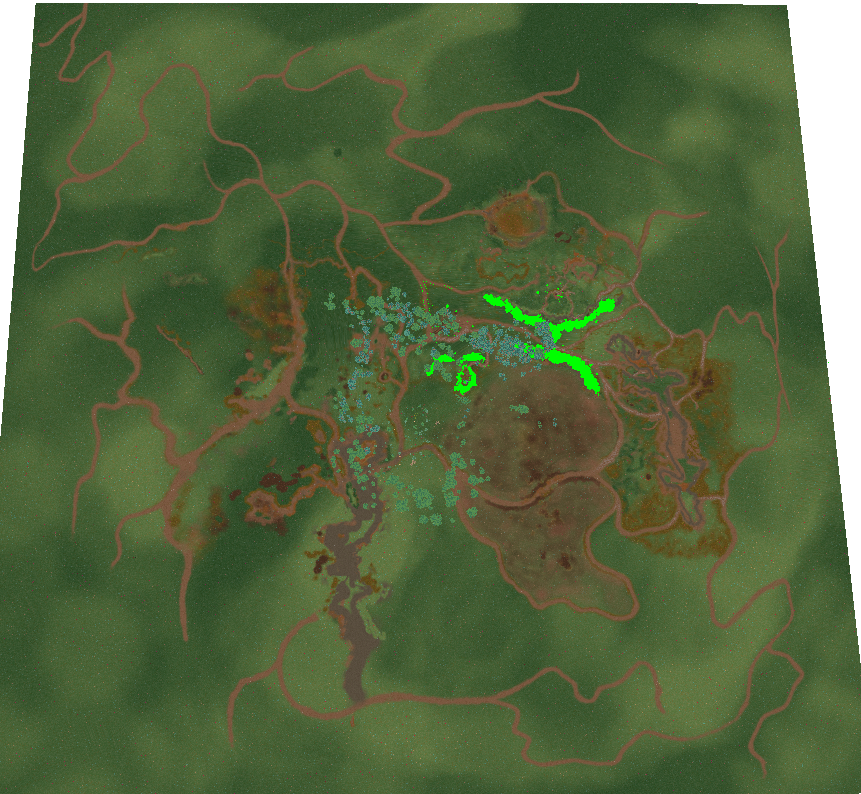
초록색 물체는 아직 터레인을 디퍼드 렌더링으로 그리지 않기 때문에 디퍼드를 껐더니 저렇게 나오게 된 것이니 큰 문제는 없다. 터레인은 얼추 잘 그려진 것 같다. 하지만 중간 중간에 검은색으로 그려지지 않은 부분이 있다. 이 Splat Map은 알파 값까지 사용하던 것이 문제였다. 애초에 Texture는 총 8개인데 r g b 3개의 채널의 이미지 2개만으로는 전부 사용이 불가능 했다. 알파 값을 사용하기 때문에 가능한 것이었다.

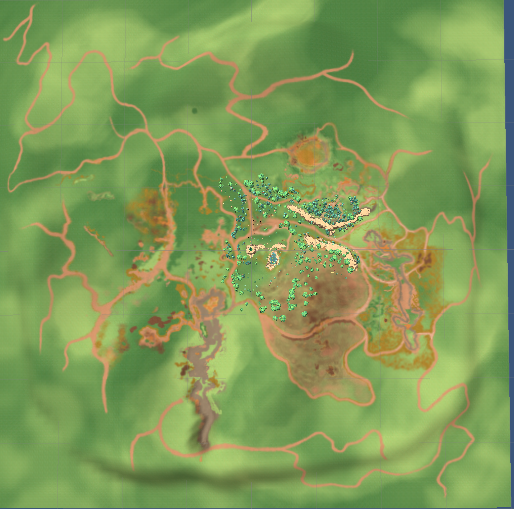
위 두 그림이 알파 값을 켜서 본 Splat Map이다. 보여지는 곳 대부분이 검은색이 보인다. 터레인에 검은색으로 구멍이 난 부분은 저 알파 값을 고려하지 않고 그렸기 때문이다. 

픽셀 쉐이더에서 각 Texture의 값을 더해주던 부분을 알파 값을 이용한 부분도 추가로 더해주었다. 0번 Splat Map은 Earth\_2, 1번 Splat Map은 Dirt\_Road를 사용한다.

이렇게 보니까 Texture의 사용 순서가 정렬 되어있다. 제작자가 만드는 과정에서 순서를 맞춘 것이 눈에 보인다.



검은색으로 구멍이 났던 부분이 사라졌다.



위 사진은 유니티 엔진에서 띄운 원본의 터레인과 구조물들이다. 빛이 아직 없어 색이 칙칙하지만 똑같이 뽑아낸 걸 볼 수 있다.

**애니메이션 블렌딩**

지금의 애니메이션은 애니메이션이 바뀔 때 뚝 끊기면서 바뀌어 부자연스럽다. 자연스러운 애니메이션을 위해 블렌딩을 구현한다.

지금은 대기 모션과 걷는 모션만 나오도록 만들어 놨다. 캐릭터의 속도를 기반으로 애니메이션이 재생되도록 만들었는데 어차피 다른 애니메이션들은 키를 눌러야 재생이 될 애니메이션들이기 때문에 애니메이션 재생하는 부분을 수정한다. Player.cpp의 MagePlayer::Update() 부분을 지워주고 GameFramework.cpp의 OnProcessingKeyboard Message 함수에서 작업한다. 함수에서 직접해도 상관은 없지만 Scene의 동명의 함수를 호출해서 Scene안에서 만들도록 하겠다.



일단 간단하게 W키를 눌렀을 때 걷는 애니메이션이 재생되고 떼면 다시 대기 모션으로 돌아가도록 했다.

SetTrackAnimationSet에서 애니메이션을 바꿔주고 AdvaceTime에서 프레임마다 애니메이션을 계산한다. 이 부분들을 수정해서 블렌딩이 되도록 한다.

어떻게 구현을 할 것인가?

1. 애니메이션이 바뀔 때 다음 애니메이션과 현재 애니메이션 둘 다 저장한다.
2. 시간에 따라 가중치를 증가하여 애니메이션이 블렌딩되게 한다.
3. 가중치가 1이 되면 완전히 애니메이션을 바꿔준다.

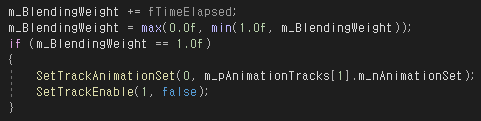
변수들을 추가해주기 전에 구조를 살펴보면 AnimationController에 AnimationTrack과 AnimationSet이 있다. AnimationTrack은 애니메이션의 재생 여부, 재생 속도, 재생할 애니메이션의 인덱스 번호 등 옵션 비슷한 값들이 들어있고 실제 애니메이션 데이터가 들어 있진 않다. Track의 인덱스 번호를 이용해 AnimationSet에 저장되어 있는 애니메이션 데이터로 애니메이션을 재생하는 구조이다.

다음 애니메이션과 현재의 애니메이션 둘 다 알아야 하므로 AnimationTrack을 2개 준비한다. 처음 모델을 생성할 때 Track을 몇 개 만들지 정할 수 잇는데 2개를 만들어 0번인덱스가 현재 애니메이션 1번 인덱스가 다음 애니메이션으로 정한다.

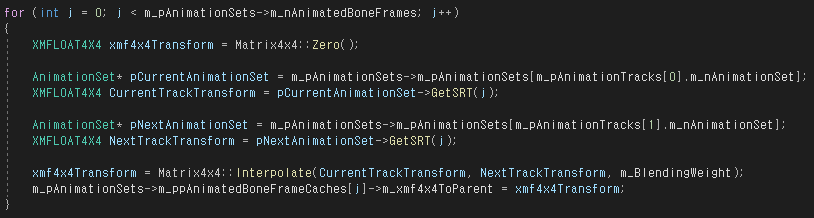


AnimationController를 생성할 때 Track을 2개 만들고 0번과 1번을 각각 세팅해준다.

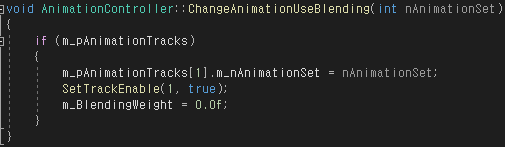
1번 Track이 활성화 상태라면 하는 상황이 아니고 아리면 0번 Track의 애니메이션이 1번의 애니메이션으로 바뀌고 있는 중인 것으로 만들려고 한다. AnimationController:: AdvanceTime()에서 활성 상태를 확인하여 if문으로 분기를 나눈다. 블렌딩을 안 하는 경우는 평소처럼 재생한다.



블렌딩을 위해 가중치를 프레임마다 점점 늘려준다. AdvanceTime 함수에는 프레임과 프레임 사이의 경과시간을 파라미터로 입력되므로 이 값을 더해준다. 가중치는 0과 1사이이므로 0과 1사이의 값으로 제한한다. 만약 가중치가 1.0이 된다면 완전히 애니메이션이 다음 애니메이션으로 바뀌었으므로 현재의 애니메이션은 0번 Track의 애니메이션을 바꿔주고 1번 Track을 비활성화 해준다.

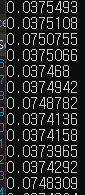


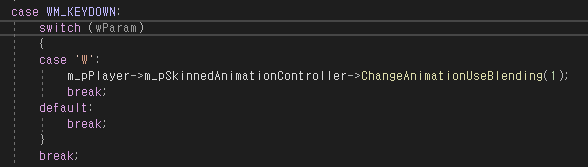
블렌딩하는 과정이다. 현재의 애니메이션 정보를 pCurrentAnimationSet에 저장하고 이를 이용해 CurrentTrackTransform에 뼈의 행렬을 저장한다. 다음 애니메이션도 마찬가지로 저장을 해준 뒤 각 뼈의 행렬을 Interpolate 함수로 보간 한다. Interpolate 함수는 각 행을 뽑아 벡터로 만든 뒤 DirectX12의 함수인 VectorLerp 함수로 보간 한 후 다시 행렬로 합쳐 반환해준다. 그렇게 보간 된 행렬을 m\_ppAnimatedBoneFrameCaches에 저장해 쉐이더로 값을 넘겨주게 된다.



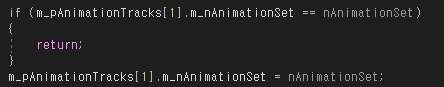
블렌딩을 하려면 1번 Track의 애니메이션을 바꿔주기만 하면 되지만 애니메이션에 관련된 모든 부분을 내가 담당하고 있으므로 나중에 팀원들이 쉽게 애니메이션을 사용할 수 있도록 ChangeAnimationUseBlending이라는 함수를 만들었다.

실행을 해보니 애니메이션이 블렌딩이 되는 것 같지만 완전히 다음 애니메이션으로 넘어가지 않는다. 모습을 보니 가중치가 자꾸 첫번째 애니메이션 쪽으로 치우쳐진 상태로 유지가 되는 것 같아서 가중치를 CMD창에 띄워 봤다.

 0.1 이상으로 올라가지 않는다. 계속 저 정도를 왔다 갔다 하다가 키보드를 떼면 그때부터 가중치가 증가해 1에서 멈춘다.



W를 누르면 애니메이션을 바꾸도록 했는데 이 함수가 계속 불리게 된다. 이 함수에서는 가중치를 0으로 초기화를 해주기 때문에 수정해줘야 한다.

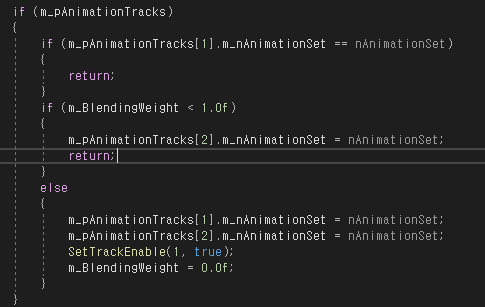


CangeAnimationUseBlending 함수에 1번 Track의 애니메이션이 바꿀 애니메이션과 같은 애니메이션인지 확인하는 코드를 추가했고 실행해보니 블렌딩이 잘 되는 것을 확인할 수 있었다.

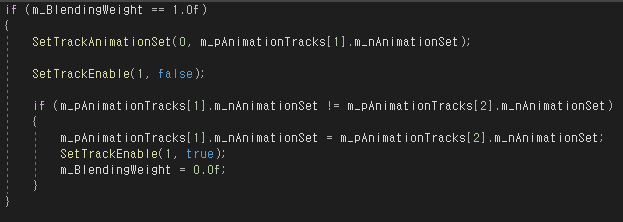
해결해야 할 문제

1. W키를 계속 누르고 있는 상태로 다른 애니메이션을 부르면 바로 걷기 모션으로 바뀜
2. 특정 애니메이션이 한번만 재생되어야 함

1번 문제는 블렌딩 되고 있는 애니메이션과 같은지를 확인하기 때문에 같은 애니메이션으로 바뀌는 것은 방지가 되지만 계속해서 걷는 중에 다른 애니메이션으로 바꾸려고 하면 블렌딩 되고 있는 애니메이션이 바뀌긴 하는데 W는 계속 눌리고 있어서 걷는 애니메이션으로 다시 바꾸려는 시도를 한다. 이를 해결하기 위해 바꿀 애니메이션을 대기시켜 놓을 3번째 Track을 만든다. 3번째 Track은 항상 비활성화 상태로 둔다. 만약 가중치가 1보다 작다면 블렌딩 진행중이고 이때 애니메이션을 바꾸게 되면 3번Track에 저장한다. 이 후 가중치가 1이 되어 블렌딩이 끝나면 3번 Track의 애니메이션을 2번 Track으로 가져오고 가중치를 다시 0으로 만들어 블렌딩이 진행되게 한다.



ChangeAnimation함수를 위와 같이 수정했다. BlendingWieght이 1보다 작을 경우 2번Track을 바꿔주고 이전에 1번 Track만 바꿔주던 것을 2번Track도 같이 바꿔주도록 했다. 2번을 바꿔주지 않으면 계속해서 2번 애니메이션으로 고정된다.

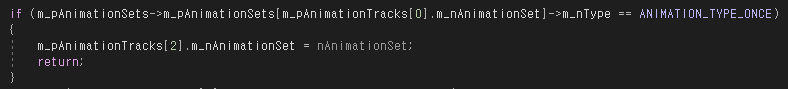


가중치가 1이 되었을 때 1번과 2번Track을 확인해 둘의 애니메이션이 다르다면 2번의 애니메이션을 1번으로 옮겨 0번과 다시 블렌딩이 되도록 했다.

2번 문제는 한번만 재생될 애니메이션의 타입 ANIMATION\_TYPE\_ONCE로 지정해서 SetPosition 할 때 따로 처리해준다. 이를 위해 SetPosition의 반환값을 void에서 bool로 바꿨고 ONCE 타입의 애니메이션이 끝나면 true를 반환하도록 했다. 값이 true가 반환되면 애니메이션을 기본 애니메이션으로 바꾸도록 만들었다.

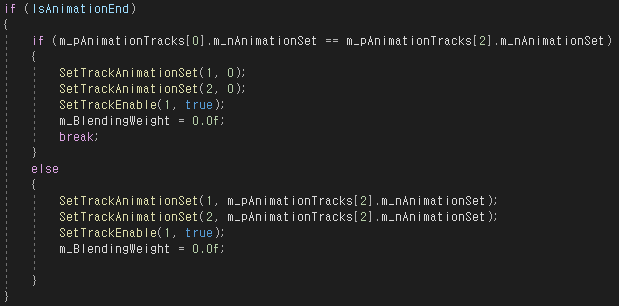
여기서도 문제가 발생했는데, 한번만 재생되는 애니메이션은 대체로 공격이나 점프, 대쉬 등 짧은 재생시간을 가지고 재생 중에 다른 애니메이션에 블렌딩 되어 변현되면 모션이 정확하게 나타나지 않아 무슨 동작인지 알기 어렵다. 실제로 걷다가 공격을 하게 되면 팔을 들어올려 공격을 하려하는 순간 걷는 모션과 다시 블렌딩 되어 내려치기 전에 다시 걷기로 돌아가 공격이 아니라 한 대 맞는 듯한 모션이 되어버린다.

애니메이션을 바꿀 때 재생중인 애니메이션의 타입에 따라 처리를 달리 해줘야 한다.



애니메이션을 바꿀 때 재생중인 현재 애니메이션인 0번 Track의 애니메이션의 타입이 ONCE이면 입시 Track인 2번 Track에 애니메이션을 저장한다.

애니메이션이 끝나서 SetPosition이 true를 반환하면 Track2번의 애니메이션을 0번Track에 적용해준다



SetPosition이 true가 될 때 0번 Track과 2번 Track이 같다면 항상 애니메이션 0번을 재생한다. 왜냐하면 저 부분이 실행되는 경우는 한번만 재생되는 애니메이션 밖에 없고 재생이 끝나서 2번 Track으로 적용을 하려 하는데 같은 애니메이션이면 그 애니메이션만 무한하게 반복하게 된다. 따라서 2번 Track이 다른 애니메이션 일 때만 그 Track의 애니메이션을 적용한다. 애니메이션 재생이 끝나면 마지막 동작에서 멈춰 있게 되고 그 동작과 바꿀 애니메이션을 블렌딩 해서 자연스럽게 다음 동작이 재생되도록 했다.