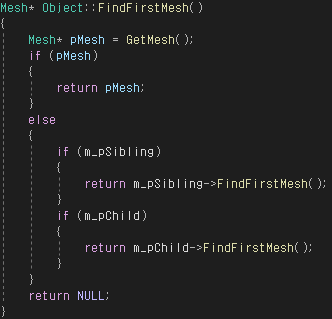
**2018180020 박재우(클라이언트) 16주차 기록**

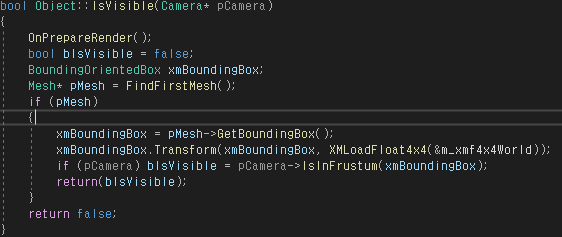
**오브젝트 컬링**

오브젝트들을 잘 불러서 그렸고 터레인도 그렸다. 하지만 그려야 할 오브젝트들이 너무 많아 프레임이 약간 떨어진다. 그래서 화면 밖 오브젝트들은 Render 함수 자체를 부르지 않도록 IsVisible 함수로 컬링을 하려 했으나 문제가 있어 수정했다.

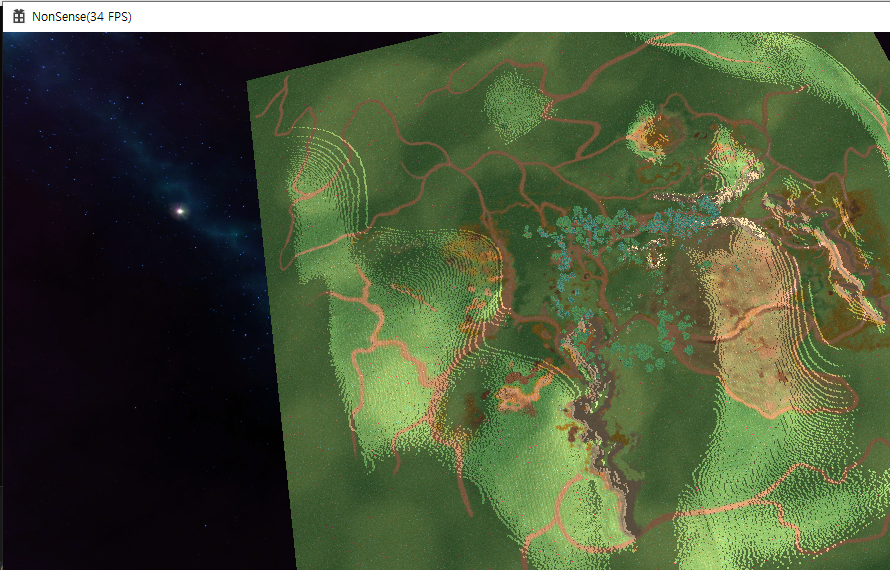
우리가 사용하는 모델은 전부 계층 구조이다. 최상위 오브젝트에는 Mesh가 없지만 만들어 놓은 IsVisible 함수는 최 상위의 Mesh의 Bounding Box를 이용해 화면 밖에 나갔는지 판단한다. 따라서 Mesh가 없다면 Sibling이나 Child에 접근해서 Mesh를 찾아야 한다.



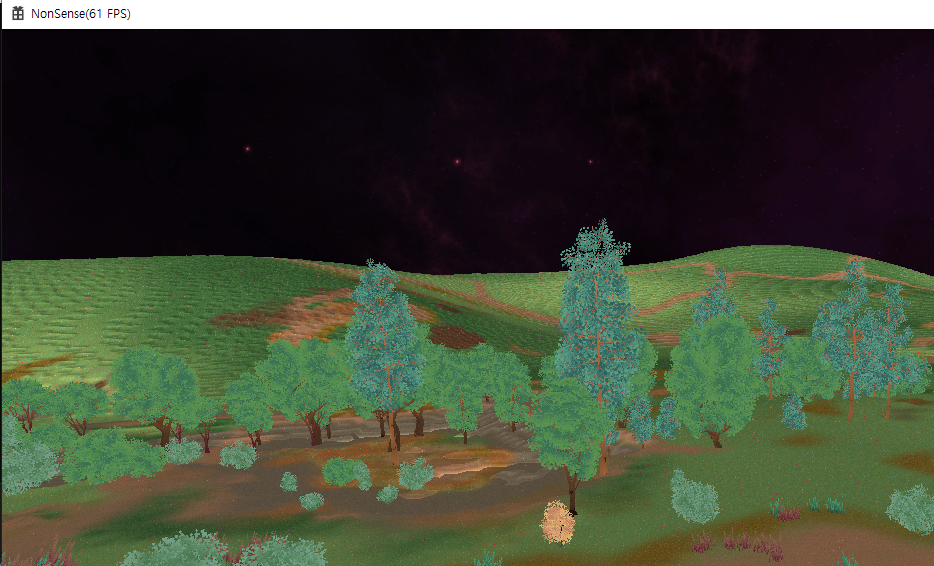
Mesh를 찾기 위해 FindFirstMesh 함수를 만들었다. GetMesh로 Mesh를 얻는데 Mesh가 NULL이라면 Sibling이나 Child에 접근해 Mesh를 찾아낸다. 가장 처음 발견된 Mesh가 있다면 그 Mesh가 반환된다.



FindFirstMesh함수를 이용해 Mesh를 Mesh의 Bounding Box를 얻어 화면 밖에 있는지 안에 있는지 확인한다. 만약 Mesh가 NULL이라면 그 객체는 Mesh가 없으므로 false를 반환해 렌더링을 하지 않도록 한다.



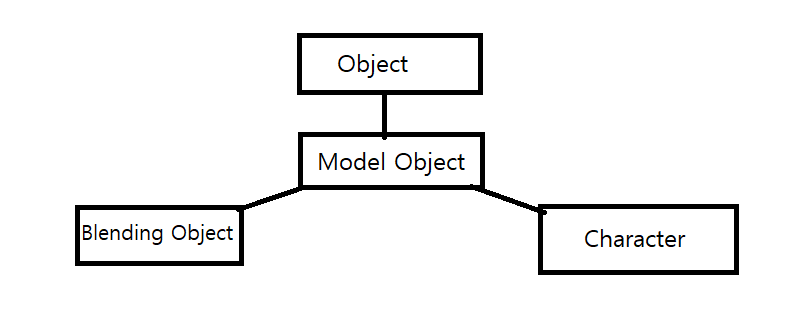
모든 오브젝트를 그릴 땐 34FPS가 나오지만



중간으로 이동해 일부 오브젝트만 바라볼 때 60FPS까지 나오는 것을 확인했다.

**클래스 구체화**

지금까지 TestModelObject 클래스로 모든 모델을 그렸었다. 이제는 실제로 모델을 움직여 각 용도별로 기능을 만들어야 하기 때문에 클래스를 나눠야 한다.



클래스는 위와 같이 만들 예정이다. 일반 구조물들은 ModelObject 클래스로 생성하고 나무와 같이 블렌딩이 필요한 오브젝트들은 BlendingObject 클래스로 생성한다. Character는 애니메이션이 필요한 스키닝 모델들을 생성할 때 사용한다. Character에서 각 종족 별 몬스터의 클래스를 만들고 컴포넌트를 만들어 추가해 줌으로써 종류를 구별한다.

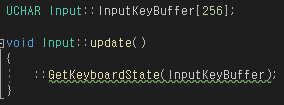
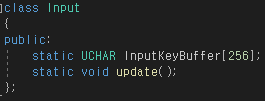
위와 같이 클래스를 만들었으나 서연이가 코드를 머지해봤더니 Blending Object는 리스트를 따로 관리해야 하지만 모든 오브젝트가 같은 리스트에 들어가는 문제가 발생하는 점을 발견해서 고쳤다.

**플레이어 조작**

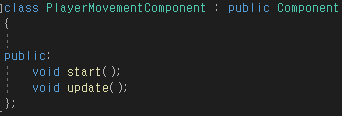
해야 할 일은 많은데 뭘 해야 할지 몰라서 회의 날짜를 앞당겼다. 구현할 기능의 우선 순위와 중간발표까지의 목표를 정하고 어느 부분을 맡을지에 대해 회의했다. 나는 플레이어 조작과 관련된 부분을 맡았다.

지금까지의 진행상황으로는 애니메이션 블렌딩이 되었고 W키(걷기), 좌클릭(공격)과 우클릭(대쉬), 스페이스바(점프)가 재생되도록 만들었다. 키보드에서 키를 떼면 기본 애니메이션으로 돌아가거나 걷는 도중 점프를 하고 나면 키입력이 동시에 2개가 안돼 애니메이션이 기본으로 돌아간다. 또 한번만 재생되는 애니메이션들은 연속으로 재생하려 하면 처음부터 다시 재생이 된다. 완전히 애니메이션이 종료될 때까지 기다려야 한다. 이 문제들을 수정하고 1인칭 시점의 플레이어 조작도 구현한다.

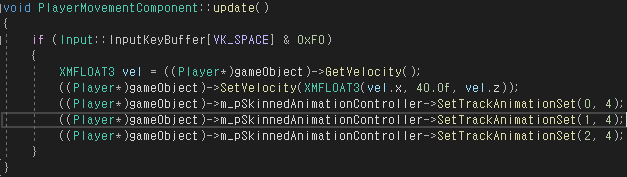
플레이어의 움직임은 컴포넌트로 제작하도록 한다. 먼저 Input 클래스를 만들어 키 입력을 받아올 수 있도록 한다.



키 입력을 받을 버퍼 Input Key Buffer를 생성한다. 그리고 update()함수에서 키 상태를 받아서 오기만 하면 되는 간단한 클래스다.



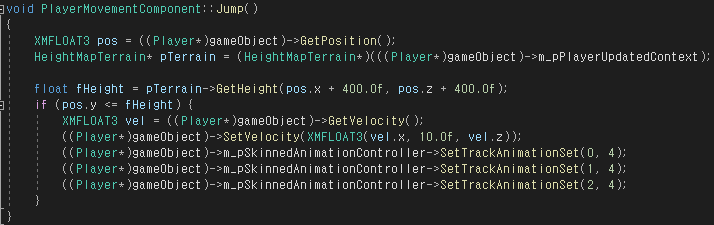
Component 클래스를 상속받는 PlayerMovementComponent를 만들었다. 지금은 아무런 변수가 없지만 플레이어의 입력이 늘어나고 필요한 부분이 생기면 점차 추가해 나갈 예정이다.



update에서 아까 Input 클래스에서 받은 키 입력을 이용해 계산을 해준다. Component 클래스에는 Object \* gameObject 변수가 있는데, 이 변수는 해당 컴포넌트를 소유중인 객체이다. 이 컴포넌트는 플레이어 객체에 사용될 컴포넌트이기 때문에 Player 클래스로 형변환을 하고 오브젝트의 y축속도를 임시로 40으로 설정했다. 그 후 점프하는 애니메이션을 재생한다. 컴포넌트를 만들었으니 만든 컴포넌트를 오브젝트에 추가해준다. 생성자에 AddCompoenet<PlayerMovementComponent>(); 를 추가해 컴포넌트를 추가해주고 Update함수에서 Object::update()를 호출한다. Object::update()에서 컴포넌트의 update 함수를 호출한다.



실행해서 스페이스바를 눌러보니 위로 튀어 오름과 동시에 점프하는 애니메이션이 재생됐다. 컴포넌트가 정상적으로 작동하는 것을 확인할 수 있었다.

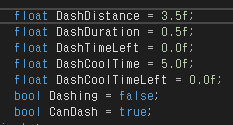


점프기능을 함수로 묶었고 현재 Y축 위치와 그 위치의 터레인의 높이를 비교해 바닥에 붙어있을 때만 점프를 할 수 있도록 제한했다.

W A S D를 눌렀을 때 애니메이션을 바꿔준다. 저번에 만든 ChangeAnimationUseBlending 함수를 이용해 애니메이션을 바꾸고 WASD 전부 눌려 있지 않은 상태라면 기본 Idle 애니메이션이 재생되도록 했다.

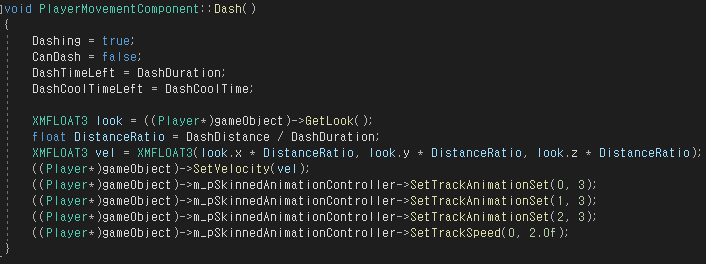
화면의 회전도 컴포넌트로 옮겼다. CursorExpos라는 Bool값을 하나 만들었고 ALT키를 누를 때마다 커서가 꺼지고 켜진다. 커서가 꺼진 상태에서 마우스를 움직이면 시야가 움직인다. 커서가 꺼졌을 때는 마우스의 커서 위치가 현재 윈도우의 중앙에서 어느정도 떨어졌는지를 계산해 그만큼 시야가 움직이게 된다. 계산이 끝나면 다시 커서를 윈도우의 중앙으로 옮겨준다.

우클릭을 했을 때 대쉬 기능을 만들었다.



대쉬를 만들기위한 7개의 변수이다. 가장 밑의 bool값들은 현재 대쉬 중인지, 현재 대쉬가 가능한지에 대한 변수이다. 쿨타임이 돌지 않았거나 대쉬 중이라면 대쉬는 불가능 할 것이다.

DashDistance는 대쉬했을 때 나아가는 거리이다. DashDuration은 대쉬가 지속될 시간이고 DashTimeLeft는 대쉬시간이 얼마나 남았는지 계산하기 위한 변수다. DashCoolTime과 DashCoolTimeLeft 변수도 쿨타임을 계산하기 위한 변수로 DashDuration, DashTimeLeft와 같은 방법으로 계산할 것이다.



Dash 함수이다. 대쉬가 실행되면 Dashing을 true로 CanDash을 false로 바꾼다. 이 두 변수는 대쉬를 사용할 수 있는지를 판단할 때 쓸 변수로 대쉬 연속 입력을 방지한다. 그 다음 DashTimeLeft를 DashDuration과 같은 값으로 맞추고 CoolTime역시 같은 방법으로 값을 세팅한다.

대쉬를 하면 특정거리를 특정시간동안 이동을 한다. 대쉬했을 때 이동할 거리와 대쉬가 지속될 시간을 나눠서 비율을 구한다. 그 비율을 플레이어의 Look벡터와 곱한 값을 속도로 넣어준다. 그 후 대쉬하는 애니메이션을 재생해준다. 여기서 대쉬 지속시간은 0.5초로 했다. 전투가 주된 컨텐츠인 우리 게임은 대쉬의 용도가 공격의 회피로 많이 쓰일 것이다. 대쉬를 이동기로는 웬만해서 사용하지 않기 때문에 짧고 빠르게 움직이는 것이 인상적이다. 애니메이션의 길이가 딱 1초이기에 1초로 하고 실행해 봤지만 1초가 생각보다 길게 느껴졌다. 그래서 0.5초로 설정했고 애니메이션이 1초의 길이이기 때문에 재생속도를 2배 빠르게 해줘야 한다. 그래서 SetTrackSpeed 함수를 사용해 애니메이션의 재생속도를 높혀준다.



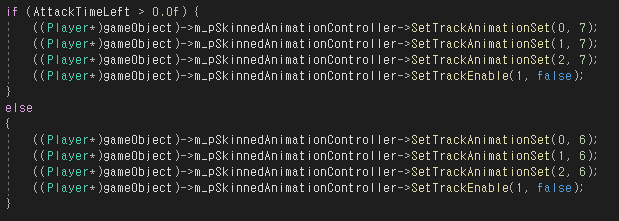
DashTimeLeft와 DashCoolTimeLeft에 프레임 경과시간을 뺀다. 이 값들이 0보다 작아지면 bool값을 바꿔준다. Dashing의 경우는 대쉬가 끝난 경우이기 때문에 플레이어의 속도를 낮추고 애니메이션의 재생속도 원래대로 만들어준다.

공격은 플레이어 조작이긴 하지만 움직임에는 관련이 없어 Movement컴포넌트가 아닌 Attack컴포넌트를 하나 만들어 사용한다.

Attack컴포넌트에 Combo Attack을 사용할지에 대한 bool 변수와 앞서 만든 대쉬처럼 공격 중인지에 대한 bool, 공격에 걸리는 시간과 남은 시간에 대한 변수가 있고 추가로 콤보를 위해 추가로 공격을 입력 받을 시간에 대한 변수도 추가했다.



Attack 컴포넌트의 update 함수이다. 왼쪽 마우스 버튼이 입력되었을 때 공격 중이 아니라면 Attack함수를 호출한다. 그리고 AttackTimeLeft의 값을 공격모션의 총 지속시간과 콤보를 위한 추가 시간을 더한 값으로 설정한다. AttackTimeLeft 값을 0 미만일 때까지 지속적으로 감소시켜주고 NextAttackInpuTime(콤보를 위한 추가 시간) 보다 적으면 During\_Attack의 값을 false로 바꿔 키 입력이 가능하게 해준다.



Attack함수에서 콤보에 관한 코드이다. 키 입력이 됐을 때 Attack이 불리는데 이 때 AttackTimeLeft의 값이 0보다 크다면 콤보 추가 시간에 키가 입력이 되었다는 뜻이기 때문에 다음 공격 모션인 7번 애니메이션을 재생하도록 한다. 마지막에 굳이 1번 Track을 비 활성화해주는 이유는 공격하는 애니메이션이 끝날 때 기본애니메이션으로 돌아가는데 이 때는 애니메이션이 블렌딩되며 바뀐다. 그래서 애니메이션이 딱 바뀌는 절묘한 타이밍에, 즉 1번 Track이 활성화 되어있는 상태에서 공격을 하게 되면 블렌딩으로 인해 자세가 이상해지게 된다. 그래서 블렌딩 대상인 1번 Track을 비 활성화해준다.