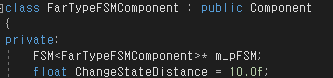
**2018180020 박재우(클라이언트) 20주차 기록**

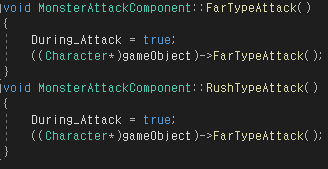
**추가 몬스터 FSM**

원거리, 돌진 몬스터의 FSM을 구현한다. FSM도 컴포넌트로 추가할 수 있게 만들었으므로 원거리와 돌진도 컴포넌트로 만들어 추가하도록 한다. 사용되는 함수들은 전부 동일하나 몇몇 함수들은 로직만 다르다. 예를 들어 근거리 몬스터가 공격할 때 플레이어와 가까워야 공격을 하지만 원거리는 먼 거리에서 공격을 시작할 것이다.

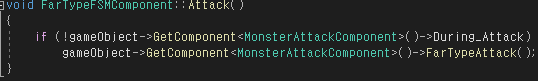


앞서 만든 FSM을 이용해 만든 FarTypeFSM에서 감지범위를 근거리보다 2배로 늘려 10.0으로 설정했다. 또한 Track 함수 안에서 사거리를 5.0으로 설정했다.

지금까지 근거리 몬스터는 플레이어와 공격방식이 동일해 같은 컴포넌트를 사용해 공격을 했지만 원거리와 돌진은 공격 메커니즘이 다르다. 따라서 Monster용 Attack컴포넌트를 만들었다. MonsterAttack 컴포넌트에 원거리와 돌진 공격을 하도록 하는 함수를 만들고 그 함수에서 Character의 해당 타입에 맞는 공격함수를 호출하도록 한다.

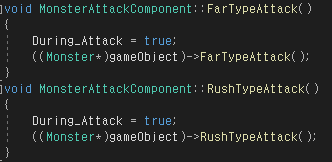


각 타입의 Attack함수를 만들었다. 앞으로 고블린, 스컬, 오크 등 여러 몬스터의 클래스가 추가되면 형변환을 할 정확한 타입을 알 수 없어지기 때문에 함수는 몬스터들의 부모인 Character 클래스에 만들었다. 지금은 함수가 호출되면 로그가 찍힐 수 있도록 해 두었고 자세한 동작은 추후에 구현한다.



FSM에서 상태에 따라 Track함수가 호출되고 사거리 안에 타겟 플레이어가 있다면 Attack함수가 호출된다. Attack함수 안에서 타입에 맞는 공격함수를 호출한다.

위와 같은 방법으로 돌진 타입의 몬스터에 사용할 FSM과 State를 만들었다.



서연이가 Character에서 Monster클래스를 자식으로 분리해서 코드를 합칠 때 충돌이 일어났다. 따라서 각 타입의 Attack함수들을 Monster클래스로 옮겼고 Character로 형변환해서 처리해 주던 부분을 Monster로 형변환을 하도록 수정했다.

로그만 찍던 Attack함수를 구체적으로 만든다. 먼저 원거리 공격부터 만들도록 한다.

공격속도는 빠르지 않으니 한번에 하나의 투사체만 날아간다. 공격을 위한 Weapon Object 클래스를 만들고 Monster가 이 오브젝트를 갖고 있도록 한다. 몬스터에 SetChild를 하지는 않는다. 프레임워크가 렌더링 할 때마다 부모의 행렬을 기준으로 위치를 계산하기 때문에 무기가 날아가는 동안 애니메이션에 의해 일직선으로 날아가지 않을 것이다. 투사체는 항상 독립적으로 존재하도록 만든다. 공격이 시작될 때 소유하고 있는 무기의 위치를 몬스터의 위치로 설정해 주고 앞으로 날아가도록 만든다. 또한 날아가는 동안에만 렌더링이 되어야 한다.

먼저 렌더링을 키고 끌 수 있도록 bool 변수를 추가한다. 변수의 이름은 Do\_Render로 한다.



Render 함수에서 시야에 있는지 체크하는 함수인 IsVisible과 함께 Do\_Render를 &&연산을 한다. 시야에도 들어와 있고 Do\_Render도 true일 때만 렌더링 될 것이다.

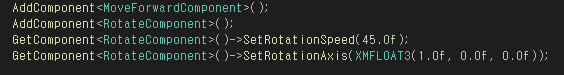
투사체를 그려줄 Object를 생성한다. WeaponObject는 만들어질 때부터 렌더링이 꺼져 있어야 한다. 생성자에서 Do\_Render를 false로 바꿔줬다.

투사체를 앞으로 날아가게 만들기 위한 컴포넌트를 만든다.

이름은 MoveForwardComponent이고 그냥 한 방향으로 움직이도록 하는 간단한 컴포넌트이다.

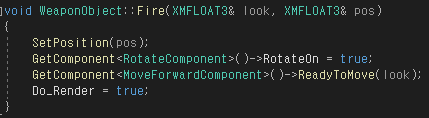


정해진 시간동안 한 방향으로 움직이도록 했고 투사체를 발사하기 전에 ReadyToMove함수를 호출해 날아갈 방향을 정하도록 만들었다. 함수로 받아올 값은 몬스터의 look벡터이다.



만든 컴포넌트를 WeaponObject 생성자에서 추가해줬다. 계속해서 회전하는 Rotate컴포넌트도 이미 만들어져 있어서 추가했다.

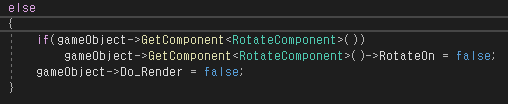
이제 MonsterAttack 컴포넌트에서 원거리 공격에서 만든 투사체를 날려주면 된다. 날려주기 위해 WeaponObject에 Fire 함수를 추가했다.



Fire에서 몬스터의 look벡터와 위치를 받아오는데 그냥 GetPosition으로 받은 위치는 밑바닥의 위치이기 때문에 y축으로 0.5높힌 위치를 넘겨줬다. 그 위치로 무기를 옮겨주고 Rotate와 MoveForward를 활성화시켜 움직이도록 하고 마지막에 Do\_Render를 true로 바꿔 렌더링 되도록 한다.



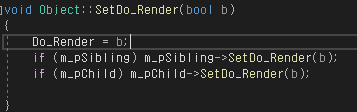
무기가 잘 날아온다. 무기는 끝까지 날아가면 그 위치에 멈춘 상태로 계속 돌고 있다. 끝에 도착하면 회전과 렌더링을 꺼주도록 한다.



MoveForward 컴포넌트에서 Rotate컴포넌트의 RotateOn과 Do\_Render를 false로 바꿔 회전과 렌더링을 비활성화 한다.

렌더링을 비활성화 했는데도 무기가 계속 그려지고 있다. 회전은 멈춘 것을 보면 코드는 정상적으로 작동했다.

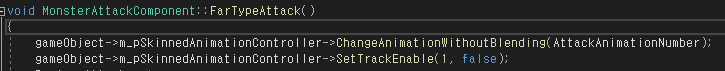
Render함수에서 if문으로 렌더링할 지 정하는데 if문 이후에 child와 sibling은 if문과 상관없이 무조건 render함수를 부르게 되어있다. 따라서 최상위 오브젝트의 Do\_render를 바꿔도 하위계층의 모든 Do\_Render를 바꿔주지 않으면 작동하지 않는 것이다.



Do\_Render를 전부 바꿔줄 수 있는 함수를 만들었고 Do\_Render를 바꿔주는 곳에 이 함수를 호출했다.

실행해 보니 잘 작동했다.

마지막으로 공격할 때 공격하는 애니메이션을 재생해준다.



공격하는 함수 처음에 애니메이션을 재생하도록 한다. 재생방법은 일반 Attack컴포넌트와 동일하다.