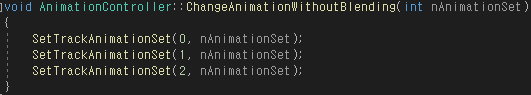
**2018180020 박재우(클라이언트) 19주차 기록**

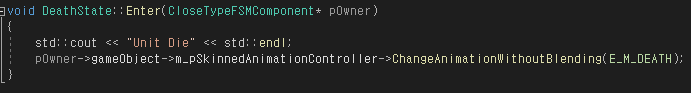
**몬스터 사망**

지금 상태는 몬스터의 HP가 0이 되어도 죽지 않는다. 0 또는 0보다 작아지면 죽는 애니메이션이 재생되면서 오브젝트가 사라지도록 만든다. 그 전에 한번만 재생되는 애니메이션으로 바꿔줄 때 모든 트랙의 애니메이션을 같은 애니메이션으로 바꿔주는 작업을 한다. 앞으로 많이 쓰일 것 같아 함수로 만들었다.



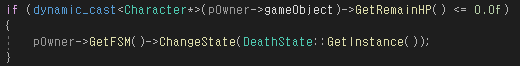
Change Animation Without Blending 함수를 이용해 죽는 애니메이션을 재생할 수 있다.

몬스터의 사망도 하나의 상태라고 생각하여 FSM으로 관리할 수 있도록 DeathState 클래스를 만들었다. 이 상태가 되면 죽는 애니메이션이 재생되고 사라지도록 만든다.



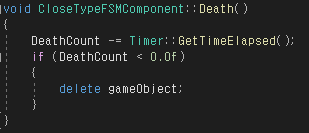
시작할 때 애니메이션을 죽는 애니메이션으로 바꿔준다. 애니메이션이 모두 번호로 관리되기 때문에 확인이 어려워 롭비가 enum으로 만들었다. E\_M\_DEATH는 몬스터의 Death 애니메이션이다.

DeathState로 변경되는 조건은 HP가 0보다 작거나 같아질 때 이므로 모든 State의 Update함수에 다음과 같은 코드를 추가했다.



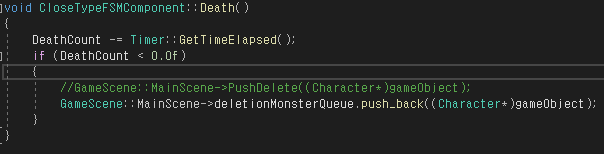
남은 HP를 검사해 조건에 맞으면 DeathState로 변경해준다. 변경이 되면 위의 Death State::Enter 함수가 호출되어 애니메이션이 죽는 애니메이션으로 바뀔 것이다.

DeathState의 Update함수는 owner의 Death함수를 호출하는 것이 다인데, Daeth함수는 시간을 계산한다.



DeathCount에서 프레임 시간을 매번 빼 주다 0 미만이 되면 gameObject를 제거한다. DeathCount는 3초로 설정했는데 death애니메이션이 3초의 길이를 갖고 있기 때문이다.

실행해 봤더니 삭제되는 순간 프로그램이 죽는다. 디버깅을 해보니 모든 컴포넌트의 update를 하고 있는 도중에 오브젝트를 지워버리니 다음 컴포넌트의 update를 하려하니 오류를 내보내며 죽었던 것이다. PushDelete라는 함수가 Delete Queue에 넣어 다음 업데이트 때 지워주는 역할을 하지만 어째서인지 제대로 작동하지 않아 직접 넣어주는 방법을 선택했다.



생으로 delete해주던 부분을 삭제를 위한 큐에 넣어주는 코드로 변경했다. 변경하고 나서 보니 삭제된 후 렌더링이 안되는 것은 확인했지만 디버깅을 해보니 오브젝트의 모든 것이 온전히 삭제되지는 않았다. Scene을 변환하면 변환할 때 마다 메모리 사용량이 계속해서 늘어났다.

코드를 확인해 보니 Object가 계층구조임에도 불구하고 소멸자에서 Sibling노드 Child노드의 Release를 부르고 있지 않았다. Material의 소멸자도 비어 있었다.