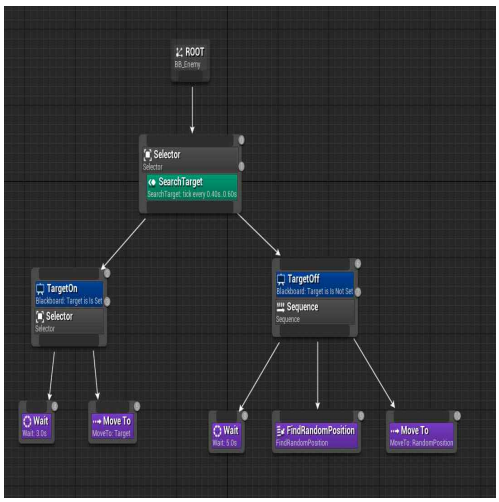
Day 10. AI 구현

10-1. 몬스터 Hp 구현

저번에 만들어놓은 캐릭터 UI를 토대로 몬스터의 Hp UI 작성하여 컴포넌트와 인스턴스를 바인드 시켰습니다.

10-2. 몬스터 AI 구현



몬스터의 움직임을 구현하기 위해 AI에 대한 코드를 작성했습니다. 전체적인 틀은 AIController와 Task, Service등은 C++코드로 만들고 값들을 담고 있는 BlackBoard와 행동을 제어할 BehavirTree는 블루프린트로 만드는 형태로 계획 했습니다. 먼저 AI 구현에 필요한 Keys 값들을 넣은 BlackBoard를 작성하여 AIController 코드에 넣었습니다. 그리고 BehaviorTree를 만들어 Selector와 Sequence의 틀만 갖춘 뒤 마찬가지로 AIController 코드에 넣었습니다. 그 다음 몬스터가 랜덤하게 움직이기 하기 위해 Task를 작성해야 했는데 BehaviorTree의 Owner의 World에서 UNavigationSystemV1 인스턴스를 가져와 GetRandomPointInNavigableRadius를 통해 랜덤한 위치 값을 얻어 BlackBoard의 TargetPosKey값에 저장시키는 방식으로 구현했습니다. 이렇게 구현한 Task는 BehaviorTree의 Sequence에 등록하여 Move to를 통해 해당 위치로 움직이게 했고 이것에 필요한 NavMesh를 맵에 설치해주었습니다.

다음으로는 몬스터가 플레이어를 찾는 Service를 구현해야 했습니다. Task로 플레이어를 찾은 후라면 BehaviorTree의 Owner를 가져와 OverlapMultiByChannel를 실행시켜 지정한 범위내 Pawn이 있는지 찾고 BlackBoard의 TargetKey값에 등록시켜 랜덤한 위치 대신 플레이어가 있는 위치로 이동하게 만들었습니다.

그런데 만들고 난 이후로도 계속 언리얼의 AI관련 기능들을 배우다보니 더 정밀하고 새로운 기술들이 있음을 알게 되었습니다. PawnSensing이라는 방식이 있고 AIPerception이라는 방식이 있는데 언리얼 커뮤니티들의 전체적인 분위기는 아무래도 신기술이었던 AIPerception을 더 선호하는 경향이 있었습니다. 지금 만든 코드도 수 시간을 들여 만든 코드라 아깝기는 하지만 다음 작업때는 AIPerception을 이용하여 다시 몬스터가 플레이어를 찾는 AI를 구현해볼 생각입니다.