**=== 네트워크 플로우 ===**

*[좀비 BT 관련 네트워크 플로우]*

* **이동 관련)**

1. [서버] 좀비 스폰 => [클라] 스폰 좀비 객체 정보 받음
2. [서버] BT 실행 -> A\* 실행 -> (경우에따라)새로운 경로, 목표지점 => [클라] 목표지점(전체 경로를 보내지 않고 바로 다음 목표지점의 좌표), 좀비 현재 위치 정보 받고 갱신
3. 경로따라 위치 갱신 & 계산 (1초씩 ?m이동)

[서버] – A\*로 나온 경로 따라 움직임 계산 (커스텀 함수 Walk 이용)

deltaTime(경과시간) 이용하기

-> 새로운 좀비 위치 클라로 전송

[클라] – 서버로부터 받은 새로운 위치로 좀비 위치 갱신 (동기화 작용)

& 서버로부터 새로 받은 목표지점으로 향하게 움직임계산 ->

서버와 똑같은 Walk 함수 만들어서 계산하기

단, 마찬가지로 딜레이 없이 deltaTime(경과시간)에

따라 위치계산하기 (언리얼 내부 함수 Tick에서)

1. [서버] BT 다시 실행 (딜레이 시간 마다) => [클라] 새로운 경로/목표지점, 좀비 위치 정보 받고 갱신

\*\*\* {결국 2번 과정으로 돌아가서 2, 3번 과정 계속 반복}

\*\*\*\*\*{결과적으로, 좀비는 서버에서 BT가 실행되는 텀마다 각 클라들에게 좀비 정보를 갱신시켜서 결국 동기화가 되는 방식}

* **BT에 Decorator에서 사용하는 bool 변수들 초기화 관련)**

해당 bool 변수 목록:

* CanSeePlayer

1. [클라]에서 “좀비 객체”가 직접 좀비 시야 검사 실시 (시야 체크는 실제로 피직스 박스에 대한 레이 캐스팅 방식으로 이루어지기 때문에 서버에서 처리하기 힘듦) => [서버] 로 검사 결과 전송 (이때, 모든 클라에서 좀비 시야 검사를 똑같이 전송하면 여러 번 서버로 같은 좀비의 시야 검사 결과를 중복해서 보내게 되므로, 해당 좀비가 포착한 해당 플레이어 클라만 해당 검사 결과를 보내도록 변경)
2. [서버] 클라에서 검사하여 검사 결과가 성공이 되어 해당 패킷을 보내온다면, 해당 좀비들의 CanSeePlayer 값 갱신
3. [서버] 해당 bool 변수값을 이용해서 BT 수행 (단, 이때 좀비가 여러 명의 플레이어를 동시에 포착했다면, 가장 가까운 플레이어를 따라가도록)

* HeardShouting, HeardFootSound, KnewPlayerLocation

1. [서버] CanSeePlayer와 달리 단순 거리 계산만으로 true/false를 판단하는 변수들이기 때문에, 서버에서 모든 판단 작업을 실행, 그 후 BT를 돌리고 다음 필요한 좀비들의 행동(task)을 각 클라들에게 전송
2. [클라]는 서버로부터 받은 task 실행

* **공격 관련)**

{좀비가 플레이어 공격}

1. [서버] BT에서 공격 범위 계산 검사

(공격 시야각 검사까지 – 공격 반경은 짧아서 Detect 시야검사처럼 장애물검사 필요X => [X] 사실 Detect에서 이미 시야 검사를 하고 여기 Attack이 실행되므로 따로 해줄 필요는 없음!)

-> 범위 안에 들면 공격 명령 전송 (전 클라이언트에게 – 애니메이션 재생 때문에)

=> [클라] 공격 명령 수신

1. [클라] 공격 수행 (공격 직접 받는 클라만 피격검사 실시) -> [클라]에서 공격 수행 후 피격 검사 수행, 결과 서버로 전송 => [서버] 피격 검사 결과 수신 -> 검사 결과 반영

[+] (클라이언트가 죽었을 때 전 클라이언트에게) => [클라] 죽은 플레이어 정보 수신

{플레이어가 좀비 공격}

1. [클라] 플레이어 공격 키 입력 -> 해당 클라 “무기 객체”에서 충돌체크 검사 => [서버] 검사 결과 전송
2. [서버] 검사 결과 반영 => 좀비 체력 동기화와 좀비 피격 애니메이션 재생을 위해 다른 [클라]들에게 해당 명령 전송
3. [클라] 다른 클라이언트들 모두 해당 좀비 체력 서버와 같게 동기화하고 피격 애니메이션 재생