**Project: UDP**

**적 AI**

본 문서에서는 게임 중 적대적 NPC에 탑재될 AI의 행동 패턴을 정의합니다. 모든 적 NPC는 이 문서에 기술된 AI를 탑재하여, 이를 기반으로 행동합니다.

**이동**

NPC는 플레이어와 일정 거리를 유지한 채 플레이어를 따라다닙니다. 플레이어보다 속도가 빠른 경우 플레이어 주변을 자전하며 다양한 방향에서 공격을 펼칩니다.

NPC도 플레이어와 마찬가지로 직진만 가능하며, 함선을 기울여 직진 방향을 변경함으로써 이동 경로를 수정합니다.

NPC는 현재 플레이어의 위치(playerPos), NPC의 위치(enemyPos), 플레이어의 이동 방향(playerMoveVector), 플레이어와의 거리(enemyDistanceMin, enemyDistanceMax)를 기반으로, 플레이어의 이동을 예상하여 이동할 위치를 계산하는 단계는 다음과 같습니다.

1. NPC는 생성 시 항상 플레이어와 유지할 최소 거리(enemyDistanceMin)와 최대 거리(enemyDistanceMax)를 무작위로 산출합니다. 두 범위는 모두 10~100 범위 내에서 산출됩니다.
2. NPC와 플레이어 사이의 거리(enemyDistanceCurrent)가 enemyDistanceMax보다 클 경우, NPC는 플레이어를 바라보려 합니다.
3. NPC와 enemyDistanceCurrent가 enemyDistanceMin보다 작을 경우, NPC는 플레이어를 바라보는 벡터의 반대 방향을 바라보려 합니다.
4. enemyDistanceCurrent가 enemyDistanceMin보다 크고, enemyDistanceMax보다 작을 경우, NPC는 플레이어의 이동 방향과 동일한 벡터를 바라보려 합니다.

NPC가 목표 방향을 천천히 바라보게 하기 위해서 Quaternion.LookRotation 함수를 사용합니다. 현재 위치에서 목표 위치를 빼서 방향 벡터를 계산한 후, LookRotation 함수를 통해 목표 Rotation 정보를 얻고, 목표 Rotation에 도달할 때까지 transform.Rotation 값을 천천히 수정하여 목표 방향을 바라보게 합니다.

**공격**

NPC는 플레이어 주변에 머물며 공격이 준비될 때마다 플레이어의 예상 위치를 목표로 공격 투사체를 발사합니다. 공격 투사체는 직선적으로 나아갑니다.

플레이어와 NPC의 위치가 계속 변화하기 때문에, 동일한 계산을 반복하여 정확도를 높입니다. 플레이어의 속도와 방향이 일정할 것을 전제하기 때문에, 플레이어가 이동 경로를 수정하여 공격을 회피할 수 있습니다.

현재 플레이어의 위치(playerPos), 이동 방향(playerDirection), 이동 속도(playerSpeed), 현재 NPC의 위치(enemyPos), NPC가 발사할 공격 투사체의 속도(enemyProjectileSpeed)를 기반으로, 플레이어의 이동을 예상하여 공격 위치를 계산하는 단계는 다음과 같습니다:

1. Vector3.distance(enemPos, playerPos) 함수를 이용하여 NPC와 플레이어 사이의 거리(betweenDistance)를 계산합니다.
2. enemyProjectileSpeed와 betweenDisatance를 나누어, NPC의 공격 투사체가 플레이어 위치에 도착하기까지의 시간(attackTime)을 계산합니다.
3. (playerPos + playerDirection) \* playerSpeed \* attackTime 수식을 이용하여 attackTime 이후 플레이어의 예상 위치(playerPredictedPos)를 계산합니다.
4. enemyPos와 playerPredictedPos 사이의 거리를 한 번 더 구하여 betweenDistance를 업데이트 합니다.
5. enemyProjectileSpeed와 betweenDistance를 한 번 더 나누어 attackTime에 업데이트 합니다.
6. (playerPos + playerDirection) \* playerSpeed \* attackTime 수식을 이용하여 playerPredictedPos를 한 번 더 계산합니다.
7. 이렇게 계산한 playerPredictedPos를 기반으로, 4~6번 단계를 한 번 더 반복하여 최종적으로 playerPredictedPos를 업데이트 합니다.
8. 최종적으로 업데이트된 playerPredictedPos를 목표로 투사체를 발사합니다. (투사체 생성 시 playerPredictedPos를 LookAt)