강화학습을 통한 회피 전략 학습

Player Slayer 프로젝트를 위한 에이전트 설계 실험

김준원

[연도]

[1. 서론 2](#_Toc199781371)

[2. Hyperparameter & Environment Parameters 3](#_Toc199781372)

[3. 보상 설계 5](#_Toc199781373)

1. 서론

1.1 프로젝트 개요 및 목적

강화학습(Reinforcement Learning, RL)은 에이전트가 환경과 상호작용하며 보상을 통해 최적의 행동 정책을 학습하는 방법론이다. 본 프로젝트는 Unity ML-Agents 툴킷을 활용하여, 에이전트가 다양한 위협 요소(해골과 유령의 탄환)로부터 생존하는 전략을 스스로 학습하도록 설계된 환경을 구축하였다.

이 프로젝트의 출발점은 기존에 기획했던 *Player Slayer*라는 게임 아이디어에서 비롯되었다. Player Slayer는 기존 Vampire Survivors라는 게임의 구조를 역발상한 게임으로, **플레이어가 직접 몬스터를 소환해 AI 에이전트를 처치하는 전략 게임**이다. 이 에이전트는 몬스터들을 처치할수록 점점 더 강해지며, 원래는 FSM(Finite State Machine) 기반의 AI로 구현되어 있었다. 하지만 해당 AI를 더욱 유연하고 전략적으로 발전시키기 위해, 강화학습 기반의 AI로의 전환을 실험하고자 이 프로젝트를 진행하게 되었다.

실제 프로젝트에서는 원래 에이전트의 복잡한 기능을 축소하고, **기본 투사체 한 종류만 자동 발사하는 상태**로 제한하여 **회피 중심의 학습 환경을** 구성하였다. 이를 통해 강화학습이 단순한 회피와 생존 전략을 얼마나 잘 학습할 수 있는지를 실험하고, 향후 Player Slayer와 같은 전략 중심 게임에서 RL 기반 AI의 활용 가능성을 탐색하고자 한다.

본 리포트에서는 실험에 사용된 하이퍼파라미터 설정과 보상 체계 설계를 설명하고, 각 보상 항목의 유무에 따른 학습 결과의 차이를 비교 분석함으로써 효과적인 보상 설계 방향에 대해 고찰한다.

1.2 실험 환경

본 실험은 Unity 엔진을 기반으로 설계된 3D 환경에서 진행되었으며, 탑다운 시점에서 상하좌우로만 이동하는 에이전트를 통해 **2차원 평면 상의 회피 행동 학습**을 다루었다. 에이전트는 고정된 y축 높이에서 이동만 가능하며, 점프, 회전 등의 복잡한 3D 움직임은 배제하였다. 이와 같은 설정은 회피 전략에 집중할 수 있도록 환경을 단순화하기 위한 의도적 제약이다.

플랫폼은 *Unity 2022.3.21f1* 버전을 기반으로 하며, 강화학습 프레임워크로는 Unity ML-Agents Toolkit이 사용되었다. Python 측 라이브러리는 *mlagents 0.30.0* 버전이 사용되었으며, Unity 패키지로는 *com.unity.ml-agents 2.1.0*버전이 적용되었다. 학습 알고리즘은 ML-Agents의 기본 PPO(Proximal Policy Optimization)를 활용하였으며, 커리큘럼 학습 기능을 통해 몬스터의 속도, 소환 주기, 수량 등을 점진적으로 조절 가능하게 구성하였다.

에이전트와 몬스터들은 Rigidbody 기반으로 물리 연산에 따라 움직이며, 최대 체력은 5로 설정되었다. 각 에피소드는 60초를 상한으로 하며, 일정 주기마다 유령과 해골이 소환되어 에이전트를 위협한다. 해골은 에이전트를 추적하며 근접 공격을 하고 유령은 에이전트를 추적하며 원거리에서 투사체를 발사한다. 에이전트는 회피 행동 외에 일정 간격으로 자동 발사되는 단일 투사체를 보유하고 있으며, 본 실험에서는 공격보다 회피 전략의 학습 성능에 초점을 맞추었다.

1. Hyperparameter & Environment Parameters

2.1 파라미터 분류 개요

프로젝트에서 사용된 파라미터는 크게 두 가지로 구분된다. 첫 번째는 강화학습 알고리즘의 학습 과정을 조절하는 하이퍼파리미터(*hyperparameter*)이며, 두 번째는 환경 구성과 난이도 조절을 위한 환경 파라미터(*environment parameters)*이다.

하이퍼파라미터는 정책 학습의 안정성과 수렴 속도에 직접적인 영향을 주는 값들로, 이번 실험에서는 대부분 ML-Agents의 기본값을 유지하여 실험 간의 일관성을 확보하였다.

반면, 환경 파라미터는 Curriculum Learning 기능을 통해 실험 도중 **동적으로 조정되도록 설계되었으며**, 에이전트가 점진적으로 강화된 환경에서 적응해 나갈 수 있도록 하기 위해 사용되었다. 해당 파라미터들은 주로 적 캐릭터의 속도, 등장 간격, 최대 등장 수 등을 제어하며, 실험의 난이도 조절 수단으로 기능하였다.

이후 섹션에서는 각 파라미터의 설정값과 적용 방식에 대해 구체적으로 서술한다.

* 1. Hyperparameter

해당 프로젝트에서는 Unity ML-Agents Toolkit에서 기본으로 제공하는 Proximal Policy Optimization (PPO) 알고리즘을 사용하였다. PPO는 연속 행동 공간에 적합하며, 구현이 간단하고 안정적인 수렴 특성을 보여 다양한 환경에서 폭넓게 사용된다. 본 프로젝트에서도 보상 구조 실험에 집중하기 위해 PPO를 선택하였다.

하이퍼파라미터는 대부분 Unity ML-Agents의 기본값을 유지하였으며, 그중 학습의 품질과 안정성에 영향을 주는 주요 항목들은 다음과 같다.

학습률(learning rate)은 ‘1.0e-4’로 설정하였다. 이는 기본값인 ‘3.0e-4’보다 작은 값인데 학습이 급격하게 흔들리는 것을 방지하고 안정적인 수렴을 유도하였다.

경험 버퍼 크기(buffer size)는 40960으로, 다양한 상황을 반영할 수 있도록 충분한 데이터를 수집하며, 배치 크기(batch size)는 2048로 설정하여 적절한 학습 속도와 안정성을 균형 있게 유지하였다.

탐색성과 정책 다양성 유지를 위한 entropy 보정 계수(*beta*)는 0.005로 설정하였다. 이는 정책의 엔트로피를 일정 수준 유지시켜, 에이전트가 특정 행동에 과도하게 집중되기보다는 다양한 움직임을 시도하도록 유도하기 위한 목적이다. 특히 회피 중심의 학습에서 다양한 전략을 탐색하는 것이 중요하므로, 탐색성을 어느 정도 확보하는 방향으로 설정하였다.

PPO 특유의 클리핑 안정성을 위한 epsilon은 0.2로 설정하였으며, 이는 일반적으로 안정적인 정책 업데이트를 위한 실험적으로 널리 쓰이는 값이다.

학습 전체에서 진행할 수 있는 최대 스텝 수는 max\_steps 항목을 통해 설정된다. 에이전트가 환경과 상호작용하며 학습할 수 있는 총 시간의 상한선을 의미하며, 일정 스텝에 도달하면 학습이 자동으로 종료된다. 초기에는 max\_steps를 1,000,000으로 설정하여 학습을 진행하였으나, 보상 구조의 복잡성과 에이전트의 학습 난이도에 비해 충분한 반복이 이루어지지 않아, 정책이 안정적으로 수렴하지 않는 현상이 나타났다. 이에 따라 학습 시간을 충분히 확보하기 위해 max\_steps를 5,000,000으로 확장하였으며, 이후부터는 각 실험에서 비교적 일관된 학습 결과가 나타나기 시작하였다.

* 1. Environment Parameters

본 프토젝트에서는 학습 난이도를 점진적으로 높이기 위해 *커리큘럼 학습(curriculum learning)* 방식을 적용하였다. 이를 위해 Unity ML-Agents의 *Environment Parameters* 기능을 활용하여, 적의 속도, 출현 주기, 수량 등을 실험 도중 자동으로 조정하도록 설정하였다.

대표적인 환경 파라미터는 다음과 같다:

* *skullSpeed*: 해골의 이동 속도는 22.0에서 시작하여, 학습 진행도(progress)가 일정 기준에 도달할 때마다 25.0, 28.0으로 점진적으로 증가하였다.
* *spawnInterval*: 몬스터가 소환되는 간격은 초기 3.0초에서 2.5초, 2.0초로 감소하여 더 자주 등장하도록 설정하였다. 에이전트가 향하고 있는 방향을 기준으로 ±30도 이내의 범위에서소환되도록 설계되었다.
* *maxSkulls*: 한 번에 소환될 수 있는 해골 수는 3에서 시작하여 최대 5까지 증가하였다.
* *bulletSpeed*: 유령이 발사하는 투사체의 속도는 40에서 시작해 최종적으로 50까지 증가하였다.

각 단계는 학습의 진행 상황을 바탕으로 자동 전환된다. 이전 조건에서 일정 수준 이상의 학습 성능이 누적되었는지를 판단하여, 소환 속도나 적의 속도를 점진적으로 조정한다. 이렇게 설정된 기준은 단순한 단기 성과가 아닌, 일정 에피소드 동안의 평균적인 학습 성과를 기반으로 판단되며, 성급한 난이도 상승을 방지하는 데 목적이 있다. 한편, 커리큘럼에 의해 변하지 않는 고정 환경 파라미터도 존재한다. 에이전트의 이동 속도는 30으로 고정되어 있다. 에이전트의 이동속도보다 해골의 이동속도가 빠를 경우 학습 난이도가 매우 어려워지기 때문에 쉽게 고착화된다. 기본 투사체는 1.2초 간격으로 자동 발사된다. 유령(Ghost)은 50의 속도로 이동하며 약 1초 간격으로 공격을 시도한다. 해골(Skull)의 체력은 3이며, 유령은 체력 2로 설정되어 있다.

1. 보상 설계

3.1 기본 보상 구성

본 프로젝트는 에이전트가 주어진 환경에서 **최대한 오래 생존하며 회피 전략을 학습하는 것**을 주요 목표로 하기 때문에, 초기부터 생존 중심의 보상 체계를 기반으로 설계되었다. 가장 기본적인 보상 구조는 다음 세 가지로 구성된다.

* 생존 시간 보상: 에이전트가 살아있는 시간 동안 지속적으로 보상을 획득하게 하여, 가능한 오래 생존하려는 행동을 유도한다. 구현에서는 일정 시간마다 일정량의 보상이 누적되는 방식으로 적용되었으며, 보상률은 초당 0.4로 설정되었다.
* 체력 감소 페널티: 에이전트가 적의 공격을 받아 체력이 줄어들 경우 즉시 페널티가 부여되며, 손실된 체력 1당 -1.0의 보상이 감점된다. 이 보상은 단순히 생존 시간만이 아닌 **실질적인 회피 성공 여부**를 판단하는 기준으로 작용한다.
* 종료 시 보상: 에피소드 종료 시점에 따라 **성공 또는 실패에 대한 명확한 보상이 주어진다.** 에이전트가 에피소드 시간을 모두 채워 생존에 성공할 경우에는 +15의 보상이 주어지며, 중도 사망 시에는 -10의 페널티가 적용된다. 이 종료 보상은 생존 그 자체의 중요성을 강조하고, 학습의 수렴 방향을 명확하게 설정해주는 역할을 한다.

이와 같은 기본 보상 구성은 실험 전반에 걸쳐 유지되며, 이후 추가되는 보상 항목들과 함께 조합되어 회피 전략 학습의 기반이 되었다.

3.2 초기 설계 시도와 실패 사례

기본 보상 구조 외에, 초기 실험 단계에서는 해골만을 대상으로 하는 단순한 보상 설계를 시도하였다. 첫 번째 방식은 가장 가까운 해골과의 거리를 기준으로 보상을 주는 구조였다. 에이전트는 매 프레임마다 가장 가까운 적과의 거리를 계산하고, 일정 거리 이상을 유지할수록 더 높은 보상을 받도록 설정하였다.

이 방식은 회피 행동을 유도할 수 있을 것으로 예상되었으나, 실제 학습 결과에서는 원하는 행동이 나타나지 않았다. 에이전트는 환경의 중심 근처에서 거의 움직이지 않고 미세하게 위치만 조정하는 행동에 고착되었으며, 결국 적에게 쉽게 사망하는 패턴이 반복되었다. 이 결과는 단일 대상과의 거리만으로는 상황의 복잡성을 충분히 반영하기 어렵다는 한계를 보여주었다.

이후에는 가장 가까운 해골로부터 반대 방향으로 이동할 경우 보상을 주는 방향 기반의 보상 구조를 실험하였다. 이 방식에서는 해골과 에이전트 사이의 방향 벡터를 계산하고, 에이전트의 이동 방향과 해골 방향 사이의 각도에 따라 보상을 차등 부여하였다. 일정 각도 이상 반대 방향일 경우에는 보상을 주고, 각도가 좁아질수록 페널티를 부과하였다.

그러나 이 방식도 안정적인 회피 행동을 이끌어내는 데 실패하였다. 방향 정보가 단기적인 움직임에 지나치게 민감하게 작용하면서, 에이전트는 흔들리는 듯한 불안정한 움직임을 반복하거나, 해골을 등지는 데 집중하여 비효율적인 경로로 이동하는 경향을 보였다. 결과적으로, 가장 가까운 해골만을 기준으로 보상을 설계하는 방식은 회피 전략을 안정적으로 학습시키기에 부족하다는 결론에 도달하였다.

이러한 시행착오를 거쳐, 이후의 보상 설계는 하나의 적이 아닌, 일정 범위 내의 위협들을 통합적으로 고려하는 방향으로 전환되었고, 현재 사용 중인 위협 밀도 기반의 보상 구조로 발전하게 되었다.

3.3 현재 보상 체계로의 전환

앞선 시도들을 통해 단일 적 기준의 보상 구조가 회피 전략을 안정적으로 유도하기 어렵다는 점이 확인되면서, 보상 설계는 다수의 위협 요소를 통합적으로 고려하는 방향으로 전환되었다. 이를 위해 도입된 개념이 바로 위협 밀도 기반 보상 구조이다.

위협 밀도는 일정 거리 이내에 존재하는 모든 적 개체와 투사체에 대해, 에이전트와의 거리 역수를 누적한 후 평균을 내는 방식으로 계산된다. 가까운 적일수록 위협 값이 커지며, 다수의 적이 근처에 있을 경우 누적 위협도가 상승하게 된다. 이 값을 일정 시간 동안 누적한 후, 에피소드 종료 시 생존 시간으로 나누어 평균 위협도를 계산하고, 그 값을 기반으로 페널티를 부여한다. 이렇게 설계된 보상은 단순히 한 대상과의 거리뿐만 아니라, 주변 환경 전체에 대한 회피 능력을 평가하는 데 적합하다.

또한 위협 밀도 외에도, 가장 가까운 해골과의 거리 변화를 추적하는 방식이 병행되어 적용되었다. 이전 스텝과 비교하여 적이 가까워졌을 경우에는 해당 거리 차이만큼 페널티가 누적된다. 단, 멀어졌을 경우에는 보상이 주어지지 않으며, 변화가 없거나 오차 범위 내에서는 무시된다. 이 방식은 단순 거리 유지보다 실질적인 거리 확보 행동을 유도하는 데 목적이 있다.

위협 밀도 기반 보상과 거리 변화 페널티는 상호보완적으로 작동하며, 에이전트가 다양한 위협을 고려해 회피 전략을 수립하고, 동시에 위험이 가까워질 때 이를 인식하고 반응하도록 학습하는 데 중점을 두고 있다.

**위협 밀도 계산 방식**

　위협 밀도는 일정 범위(40 유닛 이내) 내에 존재하는 모든 적 개체(해골)와 유령의 투사체에 대해, 에이전트와의 거리의 역수로 계산된 값을 평균하여 도출한다. 구체적으로는 다음과 같은 수식으로 표현할 수 있다.

여기서,

* ​: i번째 적 또는 탄환과의 거리
* n: 범위 내 적 및 탄환의 개수
* : 거리 0으로 인한 발산 방지를 위한 하한 설정

이 값은 일정 시간 동안 누적되며, 에피소드 종료 시 총 생존 시간으로 나누어 평균을 구한 후, 그 평균값에 음의 계수를 곱해 페널티로 작용하게 된다.

**거리 변화 계산 방식**

거리 변화는 에이전트가 가장 가까운 해골과의 거리 차를 통해 계산된다. 이전 스텝의 거리와 현재 스텝의 거리를 비교하여, 거리가 가까워졌을 경우 그 변화량을 페널티로 누적하며, 멀어졌거나 변화가 없을 경우에는 보상이 적용되지 않는다. 다음은 개념적 수식이다.

이 방식은 에이전트가 위협으로부터의 거리를 실제로 증가시킬 수 있는 행동을 유도하며, 단순히 거리를 유지하거나 정지하는 행동은 평가 대상에서 제외된다.

3.4 현재 보상 체계로의 전환

기본적인 보상 구조가 해골에 기반하여 설계된 이후, 기존 목표였던 유령을 추가하였다. 유령은 비교적 에이전트 속도보다 빠르게 움직이고 에이전트가 일정 거리내에 들어오면 투사체를 에이전트를 향해 일직선으로 발사한다.

초기에는 해골과 동일한 방식으로, 유령의 투사체 또한 가장 가까운 위협으로 판단하여 거리 변화 기반 보상에 포함시켰다. 이 방식은 이론적으로 적절해 보였지만, 실험 결과에서는 여러 가지 문제가 발생하였다. 유령이 발사하는 투사체는 짧은 시간 동안 빠르게 생성되고 사라지며, 동시에 여러 개가 존재하는 경우가 많다. 이로 인해, 매 프레임마다 계산되는 가장 가까운 위협의 위치가 불안정하게 변동하고, 그에 따른 거리 변화 페널티가 과도하게 발생하였다.

결과적으로 에이전트는 적절하게 회피하고 있음에도 불구하고 매 프레임마다 위협이 급격히 가까워지는 것처럼 인식하여 불필요한 페널티를 받는 현상이 나타났다. 이는 학습을 왜곡시키고, 오히려 에이전트가 투사체와의 거리를 안정적으로 관리하려 하기보다는 무작위적 움직임을 반복하게 만드는 결과를 초래하였다.

이러한 문제를 해결하기 위해, 투사체는 위협 밀도 계산에는 포함하되, 거리 변화 페널티 계산에서는 제외하도록 구조를 조정하였다. 즉, 해골과 투사체는 모두 일정 범위 내에 존재할 경우 위협 밀도를 높이는 데 기여하지만, 거리 변화 추적은 가장 가까운 해골과의 관계만을 기준으로 수행된다. 이를 통해 위협 인식은 보다 종합적으로 이루어지면서도, 거리 변화 기반 보상은 안정적으로 작동할 수 있도록 개선되었다.

3.5 보상 함수 구현 예시

보상 설계의 핵심 항목들은 실제 코드에서 다음과 같이 구현되었다. 이들은 모두 FixedUpdate() 내에서 매 프레임 또는 에피소드 종료 시점에 계산되어 적용된다.

**생존 시간 보상**

텍스트, 폰트, 스크린샷, 시계이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

일정 시간 동안 살아남는 것을 장려하기 위한 보상으로, 초당 0.4의 비율로 지속적으로 누적된다.

**체력 손실 페널티**

텍스트, 스크린샷, 디스플레이, 폰트이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

체력이 1 감소할 때마다 -1.0의 페널티가 적용된다. 소수점 단위 손실은 누적되며, 정수 단위로 처리된다.

**거리 변화 기반 페널티 (해골만 해당)**

**텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.**가장 가까운 해골과의 거리 차를 계산하여, 가까워진 경우에만 해당 변화량만큼 페널티를 누적한다. 미세한 변화는 무시되도록 설정되어 있다.

**위협 밀도 기반 페널티**

텍스트, 스크린샷이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.일정 거리(40 유닛) 이내의 해골과 유령 투사체를 모두 포함하여, 거리의 역수를 평균 낸 위협 밀도를 계산한다. 이 값은 일정 시간 동안 누적되며, 에피소드 종료 시 생존 시간으로 나누어 최종 페널티로 환산된다.

**텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.종료 시 보상**

에이전트가 생존에 성공할 경우 +15의 보상이 주어지며, 도중에 사망할 경우 -10을 기본으로 생존 시간에 비례한 소폭의 보상이 추가된다.

1. **보상 요소 유무에 따른 성능 비교 실험**

4.1 실험 구성 요약

이 절에서는 보상 항목에 따른 학습 성능의 차이를 비교하기 위해 설계된 실험군들의 구성을 정리하고자 한다. 본 프로젝트의 핵심 목표는 에이전트가 효과적으로 생존 및 회피 전략을 학습할 수 있도록 하기 위한 보상 구조를 찾는 것이며, 이를 위해 다양한 보상 항목을 조합하거나 제거한 실험들을 구성하였다.

기본 실험인 실험 008에서는 모든 보상 항목을 활성화한 상태로 학습이 진행되었다. 이 구성에는 생존 시간 보상, 체력 감소 페널티, 에피소드 종료 보상 외에도 위협 밀도 기반 페널티와 거리 변화 기반 페널티가 포함되어 있다. 이후의 실험들은 특정 보상 항목을 제거하여, 각 항목이 학습 결과에 미치는 영향을 비교할 수 있도록 구성되었다.

* 실험 009: 거리 변화 페널티 제거 (Dist 제외)
* 실험 010: 위협 밀도 페널티 제거 (Threat 제외)
* 실험 011: 위협 밀도와 거리 변화 페널티 모두 제거

이러한 실험 설계는, 각 보상 항목이 에이전트의 회피 전략 형성과 생존 성과에 어떤 영향을 미치는지를 구체적으로 분석하기 위한 목적으로 구성되었다.

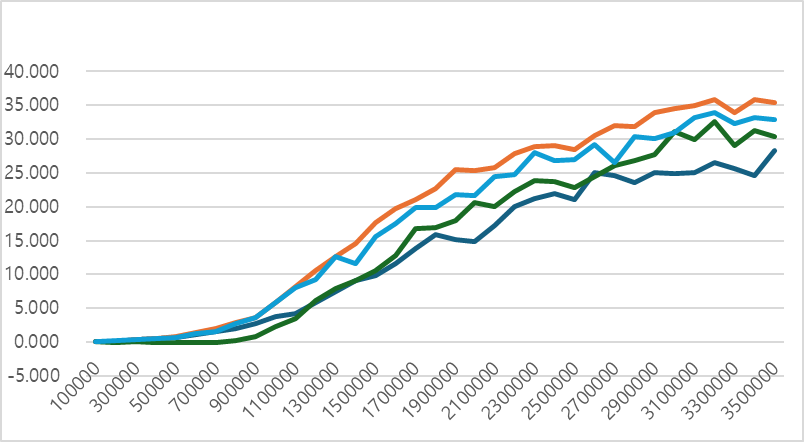
4.2 평균 보상 추이 분석

각 실험에서 에이전트가 학습을 통해 얼마나 보상을 향상시켰는지를 분석하기 위해, 평균 보상(Mean Reward)의 변화를 스텝 단위로 기록하였다. 그러나 실험마다 포함된 보상 항목이 다르기 때문에, 평균 보상의 절대값 자체는 공정한 비교 지표가 되기 어렵다. 예를 들어, 일부 실험에서는 위협 밀도나 거리 변화와 같은 페널티 보상 항목이 제거되었기 때문에, 평균 보상 값이 높게 나타나는 경향이 있다.

이를 보완하기 위해, 각 실험의 평균 보상에서 초기값을 기준으로 정렬하여 모든 실험의 시작점을 0으로 맞추고, 이후의 보상 변화량만을 비교하였다. 이렇게 정규화된 그래프를 통해 학습 효과에 의한 보상 향상 정도만을 상대적으로 비교할 수 있다.

정렬된 평균 보상 그래프를 살펴보면, 실험 009가 가장 빠르고 가파른 상승세를 보이며 최종적으로도 가장 높은 누적 보상 향상치를 기록하였다. 실험 010과 011도 비교적 빠른 상승을 보였지만, 후반부에서는 상승폭이 다소 줄어들거나 정체되는 모습을 보였다. 반면 실험 008은 모든 보상 항목을 포함하고 있음에도 불구하고, 전반적으로 가장 낮은 상승폭을 유지하며 수렴 속도도 느린 경향을 보였다.

실험 설계 초기에는 모든 보상 항목을 포함한 실험 008이 가장 우수한 성능을 보일 것으로 예상되었으나, 실제 결과에서는 거리 변화 페널티만 제거한 실험 009가 가장 뚜렷한 보상 향상 추세를 나타냈다. 이로부터, 현재의 보상 체계에 일부 비효율적이거나 과도한 항목이 포함되어 있을 가능성을 유추할 수 있다. 이에 따라 다음 절에서는 평균 보상 외에도 생존 시간, 체력 유지, 종료 성공 여부 등 보다 직접적인 성능 지표를 기반으로 각 실험 결과를 보다 정밀하게 비교하고자 한다.



<정규화된 평균 보상 추이>

4.3 상세 지표 비교

앞 절에서는 평균 보상의 변화 추이를 통해 실험별 학습 경향을 살펴보았다. 이번 절에서는 보다 세부적인 보상 항목을 기준으로, 각 실험에서 에이전트가 실제로 얼마나 성능을 향상시켰는지를 비교하고자 한다. 이를 위해 각 실험(run 008 ~ 011)에 대해, 보상 항목별로 학습 초반과 후반의 평균 값을 추출하여 시각화하였다. 여기서 말하는 초기 구간은 학습 초반 약 100만 스텝까지 수집된 보상 로그 중 가장 이른 시점 1,000개의 에피소드 평균값, 후반 구간은 전체 학습이 마무리되는 시점에서 가장 최근 1,000개의 평균값을 의미한다.

각 그래프는 보상 항목별로 초기값(파란색)과 후반값(주황색)을 나란히 배치한 형태이며, 항목은 다음과 같다:

Threat (위협 밀도 페널티), Dist (거리 변화 페널티), Survive (생존 시간), HP (체력 손실), End (종료 보상), Final (총합 보상)

각 항목은 실험 설정에 따라 포함 여부가 달라질 수 있다.

**실험 008**

실험 008은 위협 밀도 기반 페널티(Threat)와 거리 변화 페널티(Dist)를 포함한, 모든 보상 항목이 활성화된 기준 실험이다. 그래프에서 확인할 수 있듯, 위협 밀도 항목은 초반에 비해 후반에 더 작은 음수값으로 수렴하며, 비교적 명확한 개선이 이루어졌다. 이는 에이전트가 학습을 통해 근거리 위협의 밀도를 낮추는 방향으로 움직이도록 전략을 발전시켰다는 것을 의미한다. 반면, 거리 변화 항목은 학습 후반에 오히려 더 큰 음수값을 기록하고 있어, 가장 가까운 적과의 거리를 점점 줄이는 방향으로 이동하는 경향이 강화되었음을 의미한다. 즉, 위협으로부터 멀어지는 것이 아니라 가까워지는 상황이 더 자주 발생한 것이다. 이는 거리 변화 기반 보상이 실제 회피 전략을 유도하는 데 실패했거나, 보상 구조 상 해당 항목이 학습 과정에서 적절히 반영되지 못했음을 시사한다. 생존 시간(Survive)은 초기보다 크게 향상되었고, 체력 손실(HP)은 소폭 감소하였다. 종료 보상(End)도 일부 개선되었지만, 최종 보상(Final)은 여전히 음수 상태에 머물러 있다.  
이러한 결과는 에이전트가 학습을 통해 생존 전략을 일부 습득하였지만, 보상 구조 내 거리 기반 제약이 오히려 전략 형성을 방해하는 요소로 작용했을 가능성을 보여준다.

**실험 009**

실험 009는 거리 변화 페널티를 제거하고, 위협 밀도 페널티는 유지한 실험이다. 그래프에서 확인할 수 있듯, 위협 밀도 페널티 항목은 음수값이 유지되지만 학습을 통해 점차 완화되며 위협 밀도를 줄이는 방향으로 개선되었다. 이와 달리 거리 변화 페널티 항목은 보상 계산에서 완전히 제외되었기 때문에, 그래프 상에서는 0으로 표시되어 있으며 영향을 미치지 않았다. 생존 시간(Survive)은 모든 실험 중 가장 큰 폭으로 향상되었고, 체력 손실(HP) 역시 빠르게 줄어들었다.  
종료 보상(End)은 +24.49로 크게 증가하였으며, 이는 에이전트가 에피소드 종료 시점까지 생존할 확률이 높아졌다는 의미다. 최종 보상(Final) 또한 +51.85로 매우 높은 수치를 기록하였으며, 실험 전체를 통틀어 가장 우수한 결과를 보여주었다. 이러한 결과는 거리 변화 기반의 페널티가 오히려 에이전트의 학습을 방해했음을 시사하며, 해당 항목을 제거함으로써 에이전트가 위협을 피하는 데 필요한 자유로운 이동 전략을 보다 빠르게 구성할 수 있었음을 보여준다. 즉, 위협 밀도 인식은 유지하면서도 거리 기반 제약을 제거한 것이 가장 효율적인 학습 구성을 만든 실험이라고 볼 수 있다.

**실험 010**

실험 010은 위협 밀도 페널티를 제거하고, 거리 변화 페널티는 유지한 보상 구성이다. 위협 밀도 페널티 항목은 존재하지 않기 때문에 그래프에서는 0으로 표시되며, 에이전트는 학습 과정에서 위협의 밀도에 대한 정보 없이 행동을 학습하게 된다. 이 실험의 주요 특징은 거리 변화 항목이 오히려 악화되었다는 점이다. 보상 구조상 가장 가까운 적과의 거리를 멀리 유지해야 보상을 받을 수 있으나, 위협 밀도 페널티 항목이 제거되면서 에이전트는 위협 그 자체를 인식하지 못하고,  
거리만을 기준으로 판단하는 불완전한 전략을 학습했을 가능성이 있다. 결과적으로 실제 회피 행동을 잘 수행하지 못해, 거리 기반 보상은 오히려 감소하였다. 하지만 흥미로운 점은, 위협 밀도와 거리 변화 페널티를 모두 포함한 실험 008보다 실험 010의 결과가 오히려 더 우수하다는 것이다. 이 결과는 단순히 특정 항목의 유무를 넘어서, 위협 밀도 보상과 거리 변화 보상이 동시에 존재할 경우, 두 항목 사이의 상호작용이 학습에 부정적인 영향을 주었을 가능성을 시사한다. 즉, 에이전트는 두 페널티 항목을 동시에 고려하는 과정에서 혼란을 겪었거나, 상충되는 보상 신호로 인해 전략 형성이 억제되었을 수 있다. 결과적으로, 거리 기반 보상은 단독으로 존재할 때에도 부정적인 영향을 끼쳤지만, 위협 밀도 항목과 병행될 경우 더욱 효율이 감소하는 양상을 보인 것으로 해석된다.

**실험 011**

실험 011은 위협 밀도 페널티와 거리 변화 페널티 두 항목을 모두 제거한 가장 단순한 보상 구조로 구성되었다. 이 실험에서 에이전트는 생존 시간, 체력 손실, 종료 보상에 따른 직접적인 보상만을 기준으로 학습하였으며, 위협의 위치나 이동 방향에 대한 정량적인 인식 없이 회피 전략을 형성해야 했다. 그래프를 보면 위협 밀도 페널티와 거리 변화 페널티는 모두 0으로 유지되며 보상에 기여하지 않았고, 생존 시간 항목은 일정 부분 향상되었으나, 체력 손실은 전혀 개선되지 않았고 종료 보상도 낮은 수준에 그쳤다. 최종 보상 역시 네 실험 중 가장 낮은 수치를 기록하며, 학습 성과가 제한적이었음을 보여준다. 이 결과는 학습 과정에서 최소한의 위협 인식 신호조차 없는 환경이 에이전트에게 지나치게 단순하고 무의미한 보상 구조로 작용했을 가능성을 시사한다. 보상의 직접적인 기준은 존재했지만, 위협의 방향이나 밀도에 대한 유도 신호가 없었기 때문에, 에이전트는 회피 전략을 구성할 수 있는 단서를 충분히 확보하지 못했을 수 있다. 결과적으로, 실험 011은 보상의 단순화가 반드시 긍정적인 학습 결과로 이어지지는 않는다는 점을 보여주며, 적절한 제약과 정보가 함께 구성될 때에만 의미 있는 전략 학습이 가능하다는 점을 시사한다.

4.4 종합 비교 및 시사점

실험 결과를 종합해보면, 거리 변화 페널티는 단독으로 적용되었을 때는 물론, 위협 밀도 페널티와 함께 사용되었을 때에도 일관되게 부정적인 영향을 미쳤다. 오히려 두 페널티가 동시에 존재한 실험에서는 학습이 억제되거나, 방향성이 흐려지는 문제가 발생하였다. 반면, 위협 밀도 페널티만 단독으로 사용한 실험 009는 생존 시간, 종료 보상, 총합 보상 등 거의 모든 항목에서 가장 우수한 성능을 기록하였다. 이를 통해 보상 체계 설계 시, 정보의 단순화가 반드시 학습 성능을 저해하는 것은 아니며, 오히려 핵심적인 정보만을 선별하여 집중하는 것이 더 효과적인 전략 학습을 이끌 수 있음을 확인할 수 있었다. 따라서 본 프로젝트의 실제 응용 가능 보상 구조는 위협 밀도 페널티만을 남기고 거리 변화 페널티는 제거한 009 실험 방식이 가장 적절하다.

향후에는 이 구조를 기반으로 적대적 강화학습(Adversarial Reinforcement Learning) 구조로의 확장을 계획하고 있다. 기존에는 몬스터들이 에이전트의 이동 방향을 기준으로 하드코딩된 방식으로 소환되었지만, 앞으로는 몬스터를 소환하는 적대적 AI 에이전트와 이를 회피하는 학습 에이전트가 서로의 전략을 견제하며 학습하는 구조로 전환할 예정이다. 이러한 구조는 양측의 전략이 실시간으로 상호 진화하는 보다 복잡하고 현실적인 학습 환경을 제공할 것으로 기대된다.