. 인사 및 주제 소개

안녕하세요. 저는 **Crossfire 회피 전략**에 대해 발표할 박정환 입니다.
오늘 발표에서는 **Crossfire** 상황을 감지하고, 이를 회피하는 봇의 전략을 구현한 과정을 소개하겠습니다.

---

2. Crossfire란?

- Crossfire는 **두 적이 서로를 향해 있는 선분 사이**에 봇이 위치하는 상황을 말합니다.
- 이러한 상황에서는 봇이 공격받기 쉬워 생존에 큰 위협이 됩니다.
- 따라서 봇이 이를 감지하고 **안전한 위치로 이동하는 행동**이 필요합니다.

---

3. AvoidCrossfireGoal_Evaluator

- 이 클래스는 **Crossfire 회피 목표**를 설정하는 데 사용됩니다.
- 주요 역할:
    1. Crossfire 위험 상황을 평가합니다.
    2. 위험도에 따라 **안전한 위치로 이동하는 목표**를 추가합니다.

- 이 과정은 봇의 행동 우선순위를 동적으로 조정해줍니다.

---

4. 주요 메서드

(1) IsInCrossfireSituation()

- 봇이 Crossfire 상황에 처해 있는지 확인하는 메서드입니다.
- 작동 과정:
  1. 가까운 두 적을 선택합니다.
  2. 봇의 위치가 **두 적의 사격 경로에 포함되는지** 확인합니다.
  3. 거리와 위치를 기반으로 상황을 판단합니다.

(2) GetCrossfireDangerLevel()

- Crossfire 위험 상황의 **위험 수준**을 계산하는 메서드입니다.
- 계산 공식: 위험 점수=1.0−중간 지점으로부터의 거리(두 적 간 거리) / 2₩text{위험 점수} = 1.0 - ₩frac{₩text{중간 지점으로부터의 거리}}{₩text{(두 적 간 거리) / 2}}위험 점수=1.0−(두 적 간 거리) / 2중간 지점으로부터의 거리
- 점수가 1.0에 가까울수록 더 위험한 상황임을 나타냅니

다.

(3) CalculateSafePosition()

- 봇이 이동해야 할 **안전한 위치**를 계산합니다.
- 작동 과정:
    1. 가까운 두 적의 위치를 기준으로 봇이 **수직 방향**으로 이동하도록 계산합니다.
    2. 맵의 유효성을 검사하여 안전한 위치를 반환합니다.

---

5. Goal_AvoidCrossfire

- 이 목표는 봇이 Crossfire 위험에서 벗어나는 행동을 수행하도록 설정합니다.
- 주요 작업:
    1. 목표를 활성화(Activate).
    2. 하위 목표로 안전한 위치로 이동하는 명령 추가 (Goal_MoveToPosition).
    3. 위험이 해소되면 목표를 완료(completed) 상태로 변경.

---

6. 데모 영상

- 마지막으로, 봇이 Crossfire 상황을 감지하고 안전한 위치로 이동하는 과정을 담은 데모 영상을 준비했습니다.
- 이를 통해 실제 구현된 결과를 확인할 수 있습니다.

---

7. 마무리

- 이 전략은 봇의 생존 가능성을 높이고, 보다 현실감 있는 AI 행동을 구현하는 데 기여합니다.
- 질문이 있으시면 발표 후에 자유롭게 해주시길 바랍니다.
  감사합니다!