**유니티를 이용한 체스 게임 구현: 기본 체스/단색 체스에서 적용 가능한 랜덤 인공지능 및 미니맥스 알고리즘 탐구**

**1. 서론**

본 보고서는 유니티 게임 엔진을 사용하여 C# 언어로 구현한 체스 게임에 대한 개발 과정 및 결과를 소개한다. 이 프로젝트는 게임 개발 분야에서 핵심적인 역할을 하는 인공지능에 대한 이해를 깊이 하기 위해 시작되었다. 특히, 게임에서 자주 활용되는 인공지능 알고리즘 중 하나인 미니맥스 알고리즘을 탐구하고, 그 작동 원리를 시각적으로 분석하는 데 초점을 맞추었다. 본 보고서는 랜덤 이동 방식의 인공지능 구현과 더불어, 미니맥스 알고리즘의 부분적인 구현을 통해 디버그 콘솔을 이용한 시각적 분석 결과를 제시한다.

**2.1 변형 체스 탐구: 단색 체스 소개**

본 프로젝트에서는 기존 체스 규칙을 변형한 다양한 체스 변형 규칙을 탐구하였다. 탐구 결과, 단색 체스, 스파르탄 체스, almost 체스, 육각 체스, 발보 체스 등 다양한 변형 규칙을 발견할 수 있었다. 팀 내 투표를 통해 그 중 하나인 단색 체스를 게임에서 지원하기로 결정했으며 단색 체스의 규칙은 다음과 같다.

* 각 기물들이 초기에 정해진 흑색, 백색만 이동이 가능하다.
* 폰이나 룩같이 직선으로 이동이 가능한 기물은 2칸씩, 나이트는 기물을 변경하거나 한 턴에 두 번 이동 가능하며, 비숍과 퀸 같은 기물들의 대각이동은 정상적으로 이동이 가능한 체스다.
* 무승부가 날 수도 있으며 킹끼리 붙어있으면 안 되는 제약이 없는 체스다.

**2.2 게임 인공지능의 탐구: 미니맥스 알고리즘 소개**

게임 인공지능은 게임의 난이도를 조절하고 플레이어에게 흥미로운 경험을 제공하는 데 중요한 역할을 한다. 다양한 게임 인공지능 알고리즘 중 미니맥스 알고리즘은 게임 트리를 탐색하여 최적의 이동을 찾는 대표적인 알고리즘이다. 미니맥스 알고리즘은 두 명의 플레이어가 번갈아가며 최선의 수를 선택하는 게임에서, 각 플레이어가 상대 플레이어의 최선의 수를 예측하여 자신에게 가장 유리한 수를 선택하는 방식을 사용한다.

**3. 개발 환경 및 방법**

\* 개발 환경: 유니티 게임 엔진 버전: unity 2022.3.23.f1 , C#

\* 게임 구조:

\* 체스판: 유니티 화면에 직접 구현

\* 체스 기물: 프리팹으로 생성 (Chesspiece 프리팹)

\* 기물 이동: Chessman 스크립트를 통해 구현

\* 이동 가능 영역 표시: Moveplate 프리팹을 이용

\* 게임 모드:

\* m1: 사람 vs 사람

\* m2: 사람 vs AI\_1 (랜덤 이동)

\* m3: 사람 vs AI\_2 (미니맥스 알고리즘 부분 구현)

\* m4: AI vs AI (랜덤 이동, 추후 다양한 알고리즘 적용 예정)

\* 게임 진행:

\* MenuScene 씬: 단색 체스 또는 기본 체스 선택 후 ai\_select 씬으로 화면 전환

\* ai\_select 씬: m1, m2, m3, m4 중 하나 선택 후 Game 씬으로 화면 전환 후 게임 시작

**4. 인공지능 구현: 랜덤 이동 및 미니맥스 알고리즘**

**4.1. 랜덤 인공지능**

본 게임의 인공지능은 랜덤 이동 방식을 사용한다. 인공지능 턴이 시작되면, MoveBlackPiece 및 MoveWhitePiece 코루틴은 다음과 같은 과정을 통해 랜덤 이동을 수행한다.

1. 이동 가능한 기물 목록 생성: 현재 플레이어의 모든 체스말을 순회하며 이동 가능한 위치가 있는지 확인하고, 이동 가능한 체스말을 목록에 저장한다.

2. 랜덤 기물 선택: 목록에서 랜덤으로 체스말을 하나 선택한다.

3. 이동 경로 생성: 선택된 체스말의 OnMouseUp 함수를 호출하여 이동 가능한 위치를 표시한다.

4. 랜덤 이동 수행: 이동 가능한 위치 목록에서 랜덤으로 위치를 선택하고, 선택된 위치로 체스말을 이동시킨다.

**4.2. 미니맥스 알고리즘 탐구: 원리 이해**

**\* 게임 트리 탐색:** 미니맥스 알고리즘은 현재 게임 상태에서 가능한 모든 이동을 트리 형태로 나타내고, 각 노드에 대한 점수를 계산합니다.

**\* 평가 함수**: 각 노드의 점수를 계산하는 데 사용되는 평가 함수는 게임 상태를 평가하여 점수를 부여합니다. 예를 들어, 퀸은 9점, 룩은 5점, 나이트와 비숍은 3점, 폰은 1점, 킹은 1000점으로 부여하여 각 기물의 가치를 반영할 수 있습니다.

**\* 최대화/최소화 플레이어:** 미니맥스 알고리즘은 두 명의 플레이어가 번갈아가며 최선의 수를 선택하는 게임에서, 각 플레이어가 상대 플레이어의 최선의 수를 예측하여 자신에게 가장 유리한 수를 선택하는 방식을 사용합니다.

**\* 최대화 플레이어 (Maximizing Player):** 게임 결과에 대한 점수를 최대화하려는 플레이어입니다. 일반적으로 인공지능이 흑색으로 설정되어 흑색 플레이어의 입장에서 게임을 진행합니다. 흑색 플레이어는 자신의 이동을 통해 점수를 최대한 높이려고 노력합니다.

**\* 최소화 플레이어 (Minimizing Player):** 게임 결과에 대한 점수를 최소화하려는 플레이어입니다. 일반적으로 인공지능이 백색으로 설정되어 백색 플레이어의 입장에서 게임을 진행합니다. 백색 플레이어는 흑색 플레이어의 점수를 최대한 낮추려고 노력합니다.

**\* 최적의 이동 선택:** 미니맥스 알고리즘은 게임 트리를 탐색하면서 각 노드의 점수를 계산하고, 최대화 플레이어는 가장 높은 점수를 가진 이동, 최소화 플레이어는 가장 낮은 점수를 가진 이동을 선택합니다.

**4.2. 미니맥스 알고리즘 탐구: 디버그 콘솔 분석**

미니맥스 알고리즘은 게임 트리에서 최적의 이동을 찾는 인공지능 알고리즘이다. 본 프로젝트에서는 미니맥스 알고리즘을 부분적으로 구현하여 그 작동 원리를 탐구하였다. 특히 디버그 콘솔을 통해 미니맥스 알고리즘의 탐색 과정을 시각적으로 보여줌으로서 알고리즘을 더 쉽게 이해할 수 있게 구현했다.

**4.2.1. 디버그 콘솔 출력**

미니맥스 알고리즘 시각화는 m3 모드에서 확인할 수 있다.

m3 게임 모드를 선택하면 디버그 콘솔에는 미니맥스 알고리즘의 다음과 같은 정보들이 출력된다.

**\* 탐색 깊이:** "Depth: {depth}, Maximizing: {isMaximizingPlayer}" 형태로 출력된다. 예를 들어, 탐색 깊이가 2이고 최대화 플레이어(흑)의 턴이면 "Depth: 2, Maximizing: True"라고 출력된다.

**\* 가상 체스판:** "-----" 구분선으로 각 행을 구분하여 출력된다. 각 칸에는 기물의 종류를 나타내는 문자가 출력된다. 예를 들어, 흑색 킹은 "K", 백색 폰은 "p"로 표시된다. 빈 칸은 "-"로 표시된다.

**\* 가능한 이동:** "Trying move: {fromX}, {fromY} -> {toX}, {toY}" 형태로 출력된다. 예를 들어, (2, 2) 칸에 있는 흑색 나이트가 (4, 3) 칸으로 이동하는 경우 "Trying move: 2, 2 -> 4, 3"이라고 출력된다.

**\* 현재 점수:** "Current Score: {totalScore}" 형태로 출력된다. 현재 게임 상태에 대한 평가 점수를 출력한다.

**\* 최선의 이동:** "Best move for {player}: {fromX}, {fromY} -> {toX}, {toY}, Score: {score}" 형태로 출력된다. 각 깊이 레벨에서 가장 높은 점수를 가진 이동 (최대화 플레이어) 또는 가장 낮은 점수를 가진 이동 (최소화 플레이어)을 출력한다.

**4.2.2. 디버그 콘솔 분석**

디버그 콘솔을 통해 출력되는 정보를 분석하면 미니맥스 알고리즘의 작동 방식을 시각적으로 이해할 수 있다.

\* 탐색 과정: 탐색 깊이와 플레이어 정보를 통해 알고리즘이 어떤 방식으로 게임 트리를 탐색하는지 확인할 수 있다.

\* 점수 평가: 게임 상태에 대한 점수를 계산하는 방식을 파악하고, 알고리즘이 어떤 기준으로 이동을 선택하는지 이해할 수 있다.

\* 최선의 이동 선택: 각 깊이 레벨에서 가장 높은 점수 또는 가장 낮은 점수를 가진 이동을 선택하는 방식을 확인할 수 있다.

**5. UI 디자인 및 구현**

\* 메뉴 (MenuScene씬, ai\_select 씬):

\* 단색 체스 또는 기본 체스 선택 버튼 (2개)

\* m1, m2, m3, m4 게임 모드 선택 버튼 (4개)

\* 게임 씬 (Game 씬):

\* 체스판, 체스 기물 시각화

\* 게임 종료 시 승리 메시지 표시

\* UI 요소:

\* 버튼: Unity UI 시스템을 이용하여 버튼 생성 및 이벤트 처리

\* 텍스트: Unity UI 시스템을 이용하여 텍스트 표시 및 업데이트 (게임 승리 시 승리한 팀의 이름을 화면에 출력)

**6. 결과 및 분석**

\* 랜덤 이동 기반 인공지능 구현 완료

\* 미니맥스 알고리즘 부분 구현 및 탐구

\* 디버그 콘솔을 통해 미니맥스 알고리즘의 작동 과정 확인

**7. 결론 및 향후 계획**

본 프로젝트는 유니티를 활용하여 체스 게임을 구현하고 랜덤 이동 기반 인공지능을 구현하는 데 성공하였습니다. 또한, 미니맥스 알고리즘을 부분적으로 구현하여 디버그 콘솔을 통해 그 작동 원리를 탐구하고 시각적으로 분석하였습니다. 이러한 탐구 결과는 향후 미니맥스 알고리즘을 완성하고 최적화하는 데 중요한 정보를 제공할 것입니다.

\* 향후 계획:

\* 미니맥스 알고리즘 완성 및 최적화 (알파-베타 가지치기 적용)

\* 몬테카를로 알고리즘 구현 및 성능 비교

\* m4에서 다양한 인공지능 알고리즘 간 승률 비교 실험

\* 게임 UI 개선 및 사용자 경험 향상

\* 온라인 멀티플레이 기능 추가

**8.결과물**

최종 결과물:

<https://github.com/f1w2e3/Chess-AI-git/tree/merge-1-1234>

단색체스 구현;

<https://github.com/f1w2e3/Chess-AI-git/tree/monochess>

M1 구현:

<https://github.com/f1w2e3/Chess-AI-git/tree/pvp>

M2 구현:

<https://github.com/f1w2e3/Chess-AI-git/tree/test5-random-ai>

M3 구현:

<https://github.com/f1w2e3/Chess-AI-git/tree/test8>

M4 구현:

<https://github.com/f1w2e3/Chess-AI-git/tree/test9>