Zerbin Simone 5B INF

PANORAMICA DEL PROGETTO:

Il progetto Unity Chess AI mira a creare un sistema di intelligenza artificiale per giocare a scacchi utilizzando il motore di gioco Unity. L'obiettivo principale è sviluppare un robusto avversario IA in grado di giocare a scacchi contro giocatori umani con diversi livelli di abilità. Il progetto utilizza Unity versione 2019.4.15f1 ed è implementato nel linguaggio di programmazione C#.

Struttura del progetto:

Assets/: contiene tutte le risorse, gli script e le risorse del gioco.

Scripts/: contiene gli script C# per l'implementazione del sistema Chess AI.

Board.cs: gestisce la scacchiera e il suo stato

Piece.cs: Rappresenta i pezzi degli scacchi e il loro comportamento.

GameManager.cs: controlla il flusso di gioco e le interazioni del giocatore.

AIPlayer.cs: implementa la logica e il processo decisionale di Chess AI.

Scene/: Contiene le scene Unity per il gioco.

GameScene.unity: la scena principale in cui si gioca la partita a scacchi.

Prefabs/: Contiene gli oggetti prefabbricati usati nel gioco.

RAPPRESENTAZIONE DELLA SCACCHIERA:

La scacchiera è rappresentata come un array bidimensionale 8x8 nello script Board.cs. Ogni elemento dell'array rappresenta un quadrato sulla scacchiera. Lo stato della scacchiera viene memorizzato come un array multidimensionale di oggetti ChessPiece, che contengono informazioni sul tipo di pezzo e sulla sua posizione sulla scacchiera. La scacchiera viene inizializzata all'inizio del gioco e lo script Board.cs gestisce l'aggiornamento dello stato durante il gioco.

Generazione di mosse:

La generazione delle mosse è implementata nello script MoveGenerator.cs. Determina le mosse legali per ogni pezzo degli scacchi in base alla sua posizione e allo stato attuale della scacchiera. L'algoritmo di generazione delle mosse tiene conto delle regole specifiche e dei modelli di movimento di ciascun pezzo, come la promozione del pedone, l'en passant, l'arrocco e la cattura del pezzo, la presenza di un pezzo alleato o nemico. L'algoritmo genera in modo efficiente tutte le possibili mosse per ogni pezzo, prendendo in considerazione la mobilità del pezzo e le regole del gioco.

Funzione di valutazione:

L'IA valuta l'attuale posizione sulla scacchiera utilizzando una funzione di valutazione implementata nello script Evaluation.cs. La funzione di valutazione considera vari fattori per valutare l'opportunità di una data posizione. Questi fattori includono i valori dei pezzi, le posizioni dei pezzi, la struttura dei pedoni, la sicurezza del re e il controllo delle caselle centrali. La funzione di valutazione assegna un punteggio a ogni stato del consiglio, indicando la valutazione dell'IA del suo vantaggio o svantaggio.

Algoritmo di ricerca:

L'algoritmo di ricerca è implementato nello script Search.cs. Esplora le possibili sequenze di mosse analizzando in modo ricorsivo le posizioni risultanti. L'algoritmo assicura che l'IA selezioni la mossa migliore ipotizzando una giocata ottimale da entrambi i lati.

Processo decisionale AI:

L'intelligenza artificiale seleziona la mossa migliore da giocare eseguendo una ricerca limitata in profondità. Il limite di profondità determina quante mosse in avanti prenderà in considerazione l'IA. L'IA seleziona la mossa che massimizza il suo punteggio, indicando la mossa più promettente nella posizione corrente.

Interfaccia utente e interazione:

La scacchiera viene renderizzata nella scena Unity utilizzando prefabs e risorse sprite. Lo script GameManager.cs gestisce le interazioni dell'utente e gestisce il flusso di gioco. I giocatori possono selezionare e spostare i pezzi utilizzando i clic del mouse. Per assistere il giocatore, le mosse legali vengono evidenziate sul tabellone quando viene selezionato un pezzo. Le mosse possono essere visualizzate sull'interfaccia utente per fornire un feedback al giocatore.

Altro

l’Ai possiede un libro di aperture dove sono contenute numerose partite ufficiali di maestri, così da permettere al bot di giocare le prime mosse teoriche.

E’ possibile inoltre far giocare il bot contro se stesso e una volta finita una partita, è possibile salvarla dentro il file “GamesPlayed.txt”. Successivamente si può cliccare sul pulsante “Export” e esportare la partita sul famoso sito Lichess, che permette di analizzare la partita col motore.

Parti interessanti del codice

* + Per calcolare le mosse legali dobbiamo memorizzare degli offset, in particolare { 8, -8, -1, 1, 7, -7, 9, -9 }.

A screenshot of a game

Description automatically generated with medium confidence

I primi 2 indicano il movimento in verticale, il terzo e il quarto quello orizzontale e i restanti le diagonali. E’ fondamentale quindi usare gli offset in modo corretto per permettere a ogni pezzo di muoversi legalmente. Bisogna poi gestire il fatto che non possiamo continuare a muovere un pezzo se durante il suo percorso incontra un pezzo alleato, questo è gestito in questo modo:

A picture containing text, screenshot, software

Description automatically generated

Il cavallo, invece, funziona diversamente perché è l’unico pezzo che può saltare gli altri pezzi.

Vengono utilizzati questi offset: { 15, 17, -17, -15, 10, -6, 6, -10 }; Essi però non sono sempre corretti, dobbiamo gestire il fatto che potremmo trovarci sul bordo della scacchiera e quindi evitare di uscire fuori.

A picture containing text, screenshot, software

Description automatically generated

Il file PieceSquareTable.cs indica dove i pezzi sono meglio posizionati sulla scacchiera. Interessante notare come il re sia meglio ai bordi durante il mediogioco, invece meglio al centro durante il finale.

A screenshot of a computer program

Description automatically generated with medium confidenceA screenshot of a computer program

Description automatically generated with medium confidence