28/12/2017

Giovanni Sorgente Matricola: 911248

Progetto per discussione esame orale del 12/01/2018

Implementazione (mediante interfaccia grafica, libreria SDL) del gioco della Dama.

Ambiente di sviluppo utilizzato: DevC++

Schermate disponibili: menu, schermata di gioco

Menu disponibili:

1. Avvia una partita
    1.1. 1P vs 2P
    1.2. 1P vs Computer
2. Uscita dall'applicazione

Breve descrizione del progetto

Il progetto consiste nell'implementazione mediante linguaggio C del gioco della Dama.

Come descritto sopra, il software si compone di due principali funzionalità. La prima, 1P VS 2P, permette di iniziare una nuova partita contro un altro giocatore fisico, con cui si potrà disputare la sfida a turni, sempre sulla stessa macchina.

La seconda opzione (1P VS CPU) consente invece di creare una nuova sessione di gioco per sfidare un giocatore virtuale simulato interamente dal calcolatore.

L'algoritmo utilizzato per la simulazione, si basa sulla possibilità di predire le azioni dell'avversario, stabilendo una sorta di punteggio per ognuna delle mosse possibili.

Per fare un semplice esempio, immaginiamo di muovere una pedina che verrà sicuramente mangiata dall'avversario al turno successivo, a questo punto siamo in grado con molta probabilità di scartare l'opzione in favore di una mossa che sia potenzialmente meno dannosa.

Per ognuna delle pedine, si va a valutare il "peso" o "punteggio" in maniera ricorsiva di ciascuna delle mosse a disposizione.

La sommatoria dei "punteggi" attribuiti al movimento decreta l'azione da compiere per proseguire il gioco.

Cenni sulle funzionalità

Gli unici input gestiti all'interno del progetto, sono quelli provenienti da tastiera.

I tasti di direzione permettono di scorrere le voci di menu oppure di muovere il cursore (cursore arancio) all'interno della scacchiera.

Al proprio turno, un primo "invio" permette di selezionare la pedina (cursore violetto) che si intende muovere.

Occorre successivamente posizionarsi sulla cella libera di destinazione e premere nuovamente invio per terminare la mossa e spostare quindi la pedina.

In caso di mossa obbligatoria (definita all'interno del codice come "mandatoryTake"), dovuta ad una presa da effettuare obbligatoriamente, come definito nelle regole della dama italiana, le celle interessate dalla cattura saranno automaticamente colorate di verde.

Una volta catturati tutti i pezzi dell'avversario o nel caso in cui uno dei due giocatori fosse impossibilitato a compiere qualsiasi azione, il sistema stabilirà in maniera automatica il vincitore.

Spiegazione del codice

La soluzione si compone principalmente di quattro macro sezioni:

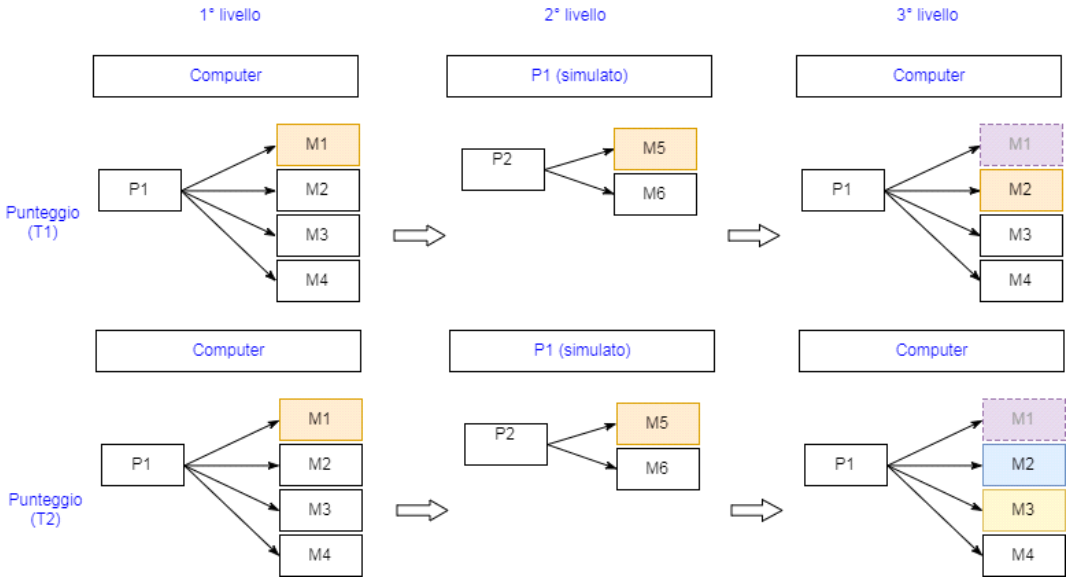
- Sessione (session.c, shared.c): il file session.c contiene tutti i metodi necessari ad inizializzare la sessione del programma, di tipo t_ApplicationSession. Quest'ultima viene creata all'avvio e detiene tutte le informazioni del modello dati necessario all'applicazione per funzionare: schermata corrente, selezione del menu utente, sessione di gioco.

  La sessione di gioco (t_GameSession) si compone di una struttura molto complessa che ha conoscenza di tutte informazioni del modello dati della partita in essere.
- Core (game_logic.c): contiene tutto l'insieme delle funzioni che hanno conoscenza di tutte le logice del programma (quali mosse sono valide, quante e quali pedine possono muovere, quali sono le "prese" obbligatorie per il giocatore di turno, ecc...)
- Rendering (render_manager.c, render_game.c): il primo file elencato contiene lo strong loop che inizializza la libreria SDL e presenta, a seconda dello stato della sessione, la schermata di gioco relativa. Si preoccupa inoltre di gestire l'input da tastiera.

  Ogni evento da tastiera cambia lo stato dell'oggetto di sessione, le modifiche così applicate verranno poi rappresentate su schermo al passaggio successivo di rendering previsto dal main loop di disegno (metodo renderScene).

  Il file "render_game" invece contiene tutto quanto attinente al disegno delle informazioni di gioco, quindi scacchiera, pedine, punteggi, ecc...

Algoritmo di simulazione del giocatore



Il grafico mostra l'idea generale che sta alla base dell'algoritmo implementato.

Supponiamo per semplicità che il player "Computer" abbia solo la pedina P1, avente la possibilità di effettuare le mosse M1, M2, M3, M4. Mentre il giocatore 1 (nella realtà giocatore fisico, ma simulato dall'algoritmo di previsione), ha a disposizione solo la pedina P2 con le mosse M5 ed M6.

Il punteggio viene calcolato per due fattori principali, la "presa" che vale 1 punto e la trasformazione in dama, alla quale viene assegnata mezzo punto.

Si calcola quindi, per ognuna delle mosse disponibili il punteggio totale delle combinazioni fino al livello di annidamento massimo definito dalla costante AI_DEEP_PATH_ANALYSIS.

Da notare che nel caso in cui la mossa sia del giocatore avversario (giocatore umano simulato) il punteggio sarà negativo.

Quindi:

- Punteggio del primo percorso: Punteggio(M1) + Punteggio (M5)+ Punteggio(M2) (NOTA, in questo caso escludiamo dal grafico M1 perché avendo effettuato la medesima mossa in origine, non sarà sicuramente più disponibile)
- Successivamente, rimanendo nel terzo livello di annidamento, calcoleremo il punteggio questa volta di M3.
- Continueremo con il procedimento di analisi per ognuno dei percorsi disponibili

Terminata l'analisi delle alberature trovate percorrendo i movimenti a disposizione delle pedine, otterremo un punteggio per ognuna delle mosse di P1.

La mossa con punteggio maggiore sarà quella scelta ed attuata dal simulatore.