Mini Laska

Gruppo 4: Giovanni Rocchini, Niccolò Simonato

Ottobre 2020-Maggio 2021

Indice

1 Introduzione e funzionamento

Questo progetto è un'implementazione di una variante del gioco Lasca.
Rispetto al gioco originale ci sono delle limitazioni sul numero di mosse per turno (max 1) e sul numero di pedine impilabili (max 3). Questa implementazione prevede le seguenti modalità di gioco:

- *1 vs. 1;*
- *1 vs. CPU.*

Il funzionamento del gioco è molto semplice: due utenti inseriscono una stringa di due coppie di caratteri alfanumerici che indicano la mossa da eseguire: i primi due caratteri contengono le coordinate della pedina da muovere, gli ultimi due le coordinate di arrivo della pedina.
Sullo schermo viene stampata la tastiera con i riferimenti delle coordinate, il numero del turno e il giocatore che deve muovere.
In caso di modalità di gioco 1 vs. CPU, il programma effettuerà in automatico la scelta della mossa.

2 Struttura progetto

2.1 Libreria

Il codice della libreria è partizionato in sezioni:

- FUNZIONI AUSILIARIE: Per interagire con le strutture dati sono state implementate delle funzioni ausiliarie.
- FUNZIONI LOGICHE DI GIOCO: Questa sezione implementa tutte le funzioni che gestiscono l'esperienza di gioco.
- GESTIONE MEMORIA: Raccolta delle funzioni che allocano e deallocano la memoria dello stato del gioco.
- FUNZIONI OUTPUT: Queste funzioni stampano sullo schermo lo stato del gioco.

2.2 pedina ADT

La pedina è definita come una struct che implementa le seguenti caratteristiche della pedina:

- Grado: un tipo enumerativo definisce se la pedina è un ufficiale o un soldato.
- Giocatore: un tipo enumerativo definisce a quale giocatore appartiene la pedina.

- Pedine catturate: due puntatori a pedine implementano le due pedine che potrebbero essere catturate.

2.3 Board

La struttura della scacchiera è una matrice bidimensionale di puntatori a variabili pedina, linearizzata. Le pedine sono posizionate nella memoria dinamica e i loro movimenti sono implementati dallo spostamento dei puntatori.

2.4 Autoplay

Autoplay è un'AI di gioco per MiniLaska, che implementa un algoritmo di tipo MiniMax per prendere decisioni sulle prossime mosse. E' implementata in un file separato dal resto della libreria, in quanto vengono definite delle ADT peculiari per il suo utilizzo:

- t-node: struct che definisce l'astrazione della mossa, ovvero i punti di inizio e fine e il valore euristico ad essa attribuito.
- t-node-list: lista di t-node, utilizzata per la decisione della mossa da effettuare.

3 Organizzazione

Di seguito sono riportate le sezioni del progetto svolte da ognuno, seguite da un breve commento dell'esperienza di sviluppo di ognuno.

3.1 Giovanni Rocchini

- Implementazione funzioni di input e gestione dei dati in ingresso.
- Implementazione funzione di gioco *isWinner()* e funzioni dipendenti.
- Sviluppo e debug di Autoplay.

Non ho incontrato particolari difficoltà nella progettazione o nello sviluppo del progetto. L'unica parte problematica è stata il debug, quando abbiamo corretto degli errori banali che compromettevano il corretto funzionamento del programma.

3.2 Niccolò Simonato

- Implementazione funzioni di gioco *move()*, *capture()* e funzioni dipendenti.
- Implementazione *main()* e funzioni dipendenti.
- Progettazione e sviluppo di Autoplay.

Lo sviluppo di questo progetto è stato agevole nel complesso. Le difficoltà che ho riscontrato sono emerse durante il debug nella fase finale di sviluppo del programma, dove una serie di errori di distrazione ha comportato dei notevoli rallentamenti alla conclusione del progetto.

4 Note di sviluppo

- Linguaggio utilizzato: ANSI C
- Documentazione generata con Doxygen
- Debug effettuato con GDB e Valgrind
- Link alla repository su GitHub:
  https://github.com/nsimonato8/MiniLaskaGruppo4.git