PROGETTO MINI LASKA

GRUPPO 51

882082 Davide Pasqual
886711 Diego Passarella
881493 Michelle Ravagnan

Struttura

Per questo progetto abbiamo organizzato diversi gruppi di funzioni in diversi file a seconda del loro scopo per garantire una lettura più ordinata ed efficace.
All'interno di src possiamo trovare:

- "colors", dove vengono create diverse funzioni per l'utilizzo di colori all'interno del terminale;
-"game_engine", dove troviamo diverse funzioni per per il gioco;
-"ia", dove vengono raccolte tutte le funzioni che riguardano la modalità "ia";
-"memory_menagement", dove sono raccolte le funzioni create per la gestione della memoria;
-''movement", dove troviamo le funzioni dedicate ai movimenti delle pedine;
-"user_interaction", dove troviamo le funzioni usate per l'interazione con l'utente.

Nelle cartelle con estensione ".h" si trovano le dichiarazioni delle funzioni.

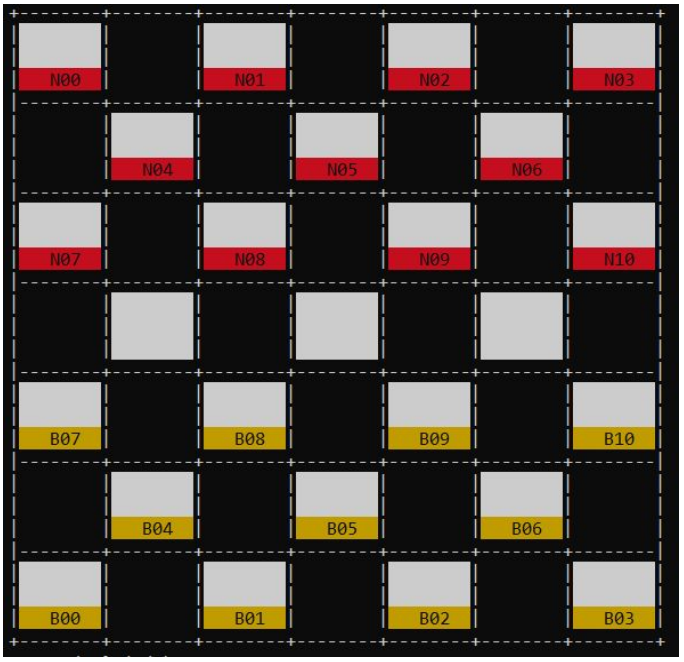
Avendo creato un makefile è possibile compilare direttamente scrivendo `make` e successivamente scrivere `./laska.exe`

La prima cosa che un giocatore vede prima di iniziare a giocare è il menu, nel quale sono presenti le seguenti opzioni:
- player vs player;
- player vs ia;
- exit.

Una volta scelta l'opzione desiderata, si potrà scegliere ulteriormente se giocare nella modalità standard oppure customizzare a proprio piacimento la grandezza della scacchiera , il numero di pedine con cui si vuole intraprendere il gioco e il colore del giocatore.

Successivamente viene presentato l'effettivo piano di gioco, la scacchiera.
La sua creazione avviene tramite le funzioni "create_board"e "print_board". Dove la prima crea la matrice bidimensionale di char e la seconda stampa il tutto aggiungendo i bordi e i colori.

Essa si presenta così (nella modalità standard):



Per la manipolazione della memoria e controllare che la memoria sia gestita correttamente abbiamo utilizzato il tool valgrind.
Per installare questo tool sono necessari i seguenti comandi da terminale:

```
sudo dpkg --configure -a
sudo apt install valgrind
```

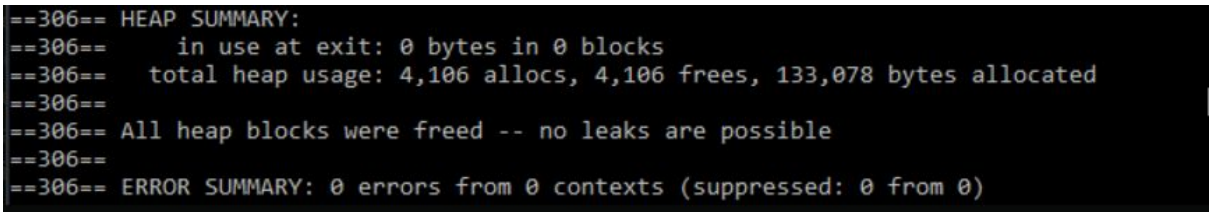
Per utilizzarlo e controllare di conseguenza la memoria abbiamo compilato il codice con le opzioni:

```
gcc main.c src/colors/colors.c src/game_engine/game_engine.c
src/ia/ia.c src/memory_management/memory_management.c
src/movement/movement.c
src/user_interaction/user_interaction.c -olaska.o -Wall -lm -g
-O0
```

E lanciato il comando:

```
valgrind -v --track-origins=yes --read-var-info=yes
--leak-check=full ./laska.o
```

In conclusione, per essere certi di aver gestito al meglio l'utilizzo della memoria, con l'aiuto di valgrind, l'output risultante è questo:



Ovvero la conferma della corretta gestione della memoria.

Divisione del lavoro

Date le circostanze il lavoro è stato compiuto principalmente online, con piccoli compiti suddivisi tra di noi mentre per le funzioni più complesse il lavoro è stato effettuato tutti insieme.

Problemi riscontrati

I vari problemi incontrati nel corso della creazione del progetto sono:

- trovare una strategia efficiente per la funzione “eat” e per il controllo delle pedine con il suo eventuale passaggio dei controlli.
- gestione coordinate delle pedine
- riadattamento funzioni di movimento per entrambi i giocatori
- comprensione algoritmo ricorsivo per la funzione “minimax”