Relazione sul progetto snake-labyrinth

Luca Menegazzo

Anno accademico 2022-2023

1 Struttura del progetto

Librerie utilizzate:
Per rendere il programma disponibile a più piattaforme possibili, sono state utilizzate *solo librerie standard già incluse in C* oppure create a mano (sempre usando standard affermati). Sarà poi il file `main.c` a chiamarle tutte.

Disposizione dei file:

- I file di codice sorgente (e labirinti pre-caricati) all'interno della cartella del progetto sono tutti reperibili fin da subito al fine di semplificare la navigazione tra file in quanto per come è stato realizzato il progetto non richiedeva una grande suddivisione strutturata dei file.
- La documentazione del progetto invece, è locata all'interno della cartella `docs` dove possiamo trovare una *documentazione tecnica* più approfondita, generata con *Doxygen*.
- È stato stilato anche un `Makefile` allo scopo di "velocizzare" l'esperienza da linea di comando, il quale permette di compilare il `main.c` con un compilatore `gcc`, attivando il flag `-Wall` oppure può essere usato per cancellare il file eseguibile prodotto dalla compilazione.

Tipi di file di codice sorgente:
Al fine di ridurre la complessità del progetto, sono stati usati semplicemente degli *header file*, risultando quindi in una *riduzione dei tempi di compilazione*.

2 Funzionalità del programma

Modalità interattiva/random:
In questa modalità al giocatore verrà chiesto come prima cosa se vuole inserire un livello creato da se stesso oppure se ne vuole selezionare uno da quelli già presenti nel file `labyrinth.txt`, dopo aver scelto il livello sarà possibile scegliere se lo si vuole giocare da sè, oppure vederlo giocato con mosse casuali.
Alla fine sarà possibile scegliere se visualizzare le mosse eseguite o meno.

Modalità challenge:
In questa modalità scarna di indicazioni accedibile tramite parametro `--challenge` da riga di comando, sarà possibile inserire il livello solo in modo manuale, verrà poi eseguito l'algoritmo usato per la risoluzione delle challenge ed infine verranno stampate a schermo le mosse eseguite dall'algoritmo.

3 Scelte di sviluppo

Strutture dati:
Le strutture dati presenti *rappresentano la maggior parte dei parametri passati alle funzioni*, in questo modo le chiamate di funzioni sono brevi e concise, inoltre è risultato più facile lavorare con tante informazioni quando quest'ultime erano raggruppate.

Algoritmo AI:
L'algoritmo usato per la risoluzione delle challenge, non sfruttando una strategia ricorsiva doveva cercare di essere meno complicato possibile, per questo non è stato possibile andare alla ricerca di tutte le monete ed evitare i vicoli ciechi.

4 Difficoltà incontrate

Challenge 2
Con il tipo di algoritmo che era stato implementato per la prima challenge, modificarlo per poter risolvere la seconda è risultato particolarmente difficoltoso, in quanto snake andava in loop cercando di trovare il modo migliore per aggirare il muro, questo problema è stato risolto tenendo conto della direzione verso cui approcciava il muro.

Coda di snake
Inizialmente cercando di evitare il problema di gestire una linked-list è stato sprecato molto tempo senza ottenere risultati implementando algoritmi più complesssi di quanto neccessario, questo ha portato ad una revisione completa della coda usando una linked-list che ha sottratto tempo al migliorare la strategia dell'algoritmo AI.