Davide Zambon (898103)  Gabriele Cusano (897835)
Alvise Spanò, Claudio Lucchese
INTRODUZIONE ALLA PROGRAMMAZIONE
2 febbraio 2023

# Snake Labyrinth

INTRODUZIONE :

L'obiettivo del progetto è implementare un gioco il cui scopo è risolvere un labirinto trovandone l'uscita. In questa relazione tratteremo i passaggi fatti per ottenere l'implementazione finale dell'intero progetto.

STRUTTURA DEL GIOCO :

Il progetto prevede che si implementino due modalità :

- Modalità interattiva :  L'utente che si approccia al gioco ha il compito di risolvere egli stesso il labirinto, muovendosi al suo interno e cercando di ottenere il punteggio migliore
- Modalità AI (intelligenza artificiale) : il programma stesso ha il compito di trovare un percorso valido che risolva il labirinto e che possibilmente sia il percorso migliore, ovvero quello che restituisce il punteggio finale più elevato.

PREMESSA E TIPI DI DATO  :

Il programma in generale inizia con la scelta della modalità. Ogni decisione dell'utente prevede che l'utente possa decidere se utilizzare una mappa predefinita oppure una  inserita da input la quale successivamente viene controllata per accertarsi che il giocatore o l'AI possa risolvere la mappa.Ogni componente del progetto è implementata attraverso l'uso delle "STRUCT" le quali sono passate  come parametri nelle varie funzioni del progetto. Le struct di maggior rilevanza sono quella del campo (*board_t*) e quella che gestisce gli array (*vector_t*), tra cui l'array che rappresenta l'avatar e quello che memorizza il percorso nella modalità AI. La prima ha come campi due variabili di tipo *size_t* che rappresentano rispettivamente il numero di righe e di colonne e un vettore di tipo *char* che memorizza la mappa. La seconda ha come campi un vettore di *int*, che nei due esempi sopra citati rappresenta la lista delle varie posizioni del corpo o delle varie mosse effettuate per uscire dal labirinto, la variabile *size* che rappresenta la lunghezza del vettore stesso (la lunghezza del corpo o la lunghezza del percorso, per esempio) e la variabile *capacity* che contiene la lunghezza massima del vettore.

MODALITÀ INTERATTIVA :

In questa modalità, il giocatore comanda il personaggio digitando in input una delle quattro direzioni proposte per muoversi : Nord, Sud, Est, Ovest. Ogni opzione corrisponde a un

carattere arbitrario scelto da noi (nello specifico: W per Nord, S per Sud, A per Est, D per Ovest). Ogni volta che l'utente inserisce il carattere corrispondente alla mossa successiva, viene controllato se il personaggio esce dal gioco, se incontra un muro e di conseguenza se ha delle trivelle per attraversarlo, se incontra bonus oppure imprevisti. Nel primo caso viene mostrato un messaggio all'utente dove viene comunicato che non può eseguire quella mossa e che ha sprecato un punto, nel secondo caso, in assenza delle trivelle, le quali danno la possibilità di attraversare il muro, viene mostrato un messaggio di avviso che non si può eseguire quella mossa e che anche in questo modo ha sprecato un punto, nel terzo caso vengono presi dei punti bonus e il personaggio di snake ha la possibilità di allungare di un'unità il suo corpo. Nell'ultimo caso invece i punti bonus vengono dimezzati e il corpo del personaggio perde metà della sua lunghezza. Come detto precedentemente, la mappa è un vettore di caratteri monodimensionale, la posizione della singola cella quindi è rappresentata da un numero, che è l'indice dell'array. Lo Snake viene rappresentato attraverso la lista delle posizioni che occupa all'interno della mappa. Riteniamo che la parte più complicata da sviluppare sia stata il movimento del corpo del personaggio, dopo che esso incontra il punto bonus oppure l'imprevisto, in quanto era necessario comprendere innanzitutto la struttura di dati da usare per rappresentare il serpente: infatti abbiamo provato a utilizzare due tipi di strutture di dati diverse tra cui le liste e il tipo *vector_t*. Inizialmente abbiamo provato ad implementare il corpo attraverso le liste che però abbiamo ritenuto poco intuitive e molto dispendiose, considerando che non vi si può accedere attraverso un indice e che per ricavare la fine del corpo è necessario scorrere tutta la lista ogni volta che si esegue una mossa. Il tipo *vector_t* invece, è stato preso di più in considerazione poiché in primo luogo la struttura è quella di un vettore e di conseguenza si accede più velocemente all'elemento desiderato in quanto non si deve scorrere la lista ma basta che venga fornito l'indice a cui il vettore deve accedere. In secondo luogo non è necessario applicare l'operazione di allocazione di memoria ogni volta che serve aggiungere un elemento nuovo al vettore ma semplicemente viene aumentato di uno il campo size e viene aggiunto l'elemento desiderato all'interno del vettore. Il movimento del personaggio innanzitutto è stato fatto pensando agli effetti che deve avere graficamente all'interno del gioco.Graficamente per ricreare la coda abbiamo ragionato riguardo alla parte di codice precedente all'effetto dell'input da parte dell'utente : prima che venga calcolata la mossa da eseguire le posizioni precedenti vengono memorizzate in un altro tipo vector_t che successivamente attraverso una funzione passa le posizioni precedenti in modo che esse risultino ordinate all'interno del corpo del serpente. Per ricreare l'operazione di allungamento abbiamo usato una funzione che aggiunge alla fine la posizione dell'ultimo elemento del vettore.Più interessante l'accorciamento nel quale si devono distinguere due casi all'interno del gioco.Il primo è quello in snake incontra un imprevisto,il secondo è quello in cui snake incontra un pezzo del suo corpo.In entrambi i casi sono state usate la stessa funzione che calcola la nuova lunghezza in base all'indice che gli si dà in input.

MODALITÀ INTELLIGENZA ARTIFICIALE:

In questa modalità, l'utente ha l'unico compito di scegliere quale mappa usare, è poi il computer che, nel limite massimo di un minuto, deve trovare il percorso migliore (o quello che più gli si avvicina). La sfida maggiore riscontrata è stata da subito il trovare un algoritmo che decifrasse la struttura della mappa per trovare una soluzione. Il primo metodo implementato è stato quello suggerito alla consegna, ovvero il *"sempre a destra"*: tenendo un muro sempre sulla

destra, è matematico che prima o poi si raggiunga l'uscita. La difficoltà in questo caso era nel concetto di "destra", che è relativo e non assoluto. La soluzione è stata trovata creando prima il concetto di "direzione", dipendente dall'ultimo passo eseguito. È una soluzione valida ma poco efficace, e non tiene conto delle celle speciali ($, !, T). Una soluzione migliore è quella che abbiamo nominato "*numeretti*" nel codice sorgente. A partire dalla cella iniziale, a cui si dà valore 1, si assegna valore 2 alle celle vuote adiacenti, e così via, ogni volta si incrementa di uno. Così facendo, si arriva alla fine col numero di passi minore in assoluto, ma continuano a non essere prese in considerazione le celle speciali. La soluzione finale che abbiamo pensato è una funzione ricorsiva, che ad ogni passo esplori tutti i possibili percorsi nelle quattro direzioni, calcoli per ognuno il punteggio e restituisca il migliore. Questa modalità tiene conto di tutto e restituisce il percorso migliore, ha però il difetto dell'inefficienza computativa (al più quattro chiamate ricorsive per ogni passo), che richiede tempi enormi per la risoluzione. Abbiamo trovato due soluzioni a questo problema. La prima elimina parte della difficoltà computazionale modificando la mappa tramite euristiche (quali per esempio riempire di muri i vicoli ciechi o i passaggi di larghezza maggiore di 1) che intaccano solo il funzionamento del trapano e solo in piccola parte, dando quindi un enorme vantaggio a fronte di una penalizzazione statisticamente bassa sul punteggio finale. La seconda invece non risolve il problema ma lo tronca. Al raggiungimento del minuto, viene restituito il percorso migliore finora trovato. Vista comunque l'efficacia ed efficienza della soluzione "numeretti", abbiamo deciso di tenerla, per restituire all'utente prima una soluzione veloce, per poi esplorare ricorsivamente fino al raggiungimento del minuto (o della soluzione) e confrontare la miglior soluzione tra le due trovate.

## DIVISIONE DEL LAVORO

Per la realizzazione del progetto, abbiamo alternato momenti di programmazione sincrona a quattro mani con momenti di divisione dei compiti, per rendere più efficiente lo sviluppo modulare, in modo che le modifiche di uno non intaccassero né dipendessero dal codice dell'altro, e viceversa. Abbiamo iniziato creando insieme la struttura "contenitore" del progetto (creazione della mappa e logica del main), le funzioni comuni (come per esempio la lettura in input della mappa personalizzata), una prima bozza di funzionamento della modalità interattiva e della modalità AI col sistema "sempre a destra", e le challenge intermedie, tutto ciò con le conoscenze del linguaggio apprese nelle prime fasi del corso. Successivamente i compiti si sono divisi in modo più marcato, la modularità della struttura ha permesso di continuare lo sviluppo delle due modalità separatamente e in modo indipendente, per quanto comunque abbiamo continuato a passarci il codice e a confrontarci sulle soluzioni trovate. È stato bello vedere come la scrittura pratica del codice possa essere indipendente anche se le soluzioni teoriche trovate sono sempre frutto del confronto e dello scambio di idee. In particolare nello sviluppo dell'intelligenza artificiale, la stesura del codice è solo l'1% della difficoltà, che risiede totalmente nella progettazione degli algoritmi, che sono sempre stati frutto di ragionamenti condivisi.