Relazione progetto Snake Labyrinth

Realizzato da:
Luca Barban, Leonardo De Zottis, Paolo Mozzoni e Luca Saccarola

Introduzione

Per risolvere la parte pratica dell'esame di introduzione alla programmazione ci è stato chiesto di ricreare una versione del famoso gioco snake del 1976. Nella nuova versione il nostro serpente deve affrontare un labirinto e arrivare alla fine con il minor numero di mosse possibili. All'interno del labirinto sono presenti vari oggetti che permettono di migliorare o peggiorare il punteggio finale. La consegna prevedeva di creare due modalità: la prima doveva essere interattiva, quindi una modalità che prevede la diretta interazione del giocatore con l'ambiente di gioco; la seconda invece doveva far svolgere il labirinto ad un algoritmo di risoluzione, questo poteva fare due cose o procedere verso l'uscita tenendo un muro sempre a destra, oppure doveva trovare l'uscita con delle mosse casuali.

Specifiche del progetto

Il progetto che abbiamo composto in accordo con la consegna contiene quattro diverse modalità: la prima modalità è interattiva e è divisa in cinque livelli di difficoltà crescente, la mappa è composta da dei muri rappresentati con il carattere "#" e dentro ogni labirinto sono presenti dei trapani "T", che consentono al giocatore di rompere tre pareti, aprendo un nuovo passaggio, insieme ai trapani l'altro oggetto che conferisce un bonus sono le monete "$" le quali permettono a snake di allungarsi e aggiungono 10 punti per ogni moneta. L'imprevisto "!" è l'unico oggetto che ostacola il giocatore, infatti una volta raccolto va a dimezzare il numero di monete raccolte. L'obiettivo del gioco non consiste semplicemente nel trovare l'uscita del labirinto ma nel raggiungerla con il punteggio maggiore, infatti il giocatore comincerà la partita con un totale di 1000 punti e ad ogni passo ne perderà uno, il punteggio finale quindi si ottiene aggiungendo al punteggio iniziale 10 punti per ogni moneta e togliendo un punto per ogni passo eseguito. La modalità automatica seguirà le medesime regole solo che in questo caso a muoverlo lo snake non sarà il giocatore ma degli algoritmi che in base alla modalità svolgono funzioni differenti. Le modalità automatiche sono tre: la prima che prevede l'esecuzione di mosse casuali fino al raggiungimento dell'uscita; la seconda modalità fa in modo che snake abbia sempre un muro alla propria destra, permettendogli così di raggiungere l'uscita. La terza e ultima modalità automatica che abbiamo implementato prevede l'ottimizzazione del percorso facendo arrivare snake all'uscita attraverso il percorso più rapido e permettendogli di ottenere il punteggio maggiore possibile.

Sviluppo del progetto

Per lo sviluppo del progetto abbiamo diviso il lavoro in: sviluppo della mappa, progettazione degli algoritmi per le risoluzioni, sviluppo dell'interfaccia grafica e documentazione. Il primo punto che abbiamo sviluppato è stata la costruzione della mappa, che è sviluppata nei file *map*, all'interno dei quali abbiamo definito gli elementi della mappa, ossia muri, monete, trapani, imprevisti e spazi attraversabili. Inoltre sono presenti funzioni che ci permettono di caricare la mappa in diversi modi: il primo è quello di inserire manualmente i valori, il secondo ci permette di caricare la mappa da un file di testo.
Il secondo punto che abbiamo sviluppato sono gli algoritmi di risoluzioni; il più complesso è quello che ci permette di risolvere il labirinto nel modo più efficiente possibile, questo è contenuto nei file denominati *backtracking*, dove sono contenute le varie funzioni per trovare lo scenario ottimale e dare il giusto peso ad ogni elemento presente nella mappa. Gli altri due algoritmi di risoluzione sono risultati più semplici da realizzare, infatti nei file *random* troviamo le funzioni che fanno muovere il nostro snake in maniera casuale, mentre nei file *right* sono contenute quelle che gli permettono di avere sempre il muro sulla destra. Tutte queste funzioni inviano le mosse e il loro percorso ai file denominati *path* che si occupano di stampare ed eseguire la combinazione.
L'ultima fase di scrittura del codice ci ha fatto sviluppare l'interfaccia del giocatore, quindi gli aspetti legati alla forma di snake, la cui gestione è affidata ai file *queue*, questi gestiscono l'allungamento e accorciamento della coda, infatti nel caso in cui si passa sopra una moneta snake si allungherà, mentre se incontra un imprevisto la sua lunghezza sarà dimezzata. Oltre a snake abbiamo curato anche l'interfaccia grafica del menù che viene gestita dai file *ui* i quali tengono conto anche degli errori che il gioco può riscontrare, come l'inserimento di un carattere non consentito nella generazione della mappa, oppure la mancata assegnazione della memoria.
Una volta completato il codice abbiamo eseguito tramite doxygen la documentazione andando a dare una definizione delle varie funzioni create, specificando il loro funzionamento e allegando grafici per le chiamate che ogni funzione compie.

Elementi teorici utilizzati

Per completare questo progetto abbiamo sfruttato diverse funzioni presenti nella libreria standard di C99. Siamo andati a definire delle strutture dati, quindi abbiamo creato dei tipi di dati che possono contenere più variabili al loro interno e usando la funzione *typedef* ne abbiamo dato una definizione, ossia abbiamo attribuito a queste strutture delle parole chiave per richiamarle. Oltre alle strutture dati abbiamo dichiarato anche variabili di tipo *int* ossia numeri interi e *char* che prevede un carattere.
Abbiamo usato la funzione *malloc* che ci permette di assegnare una certa quantità di memoria ad un puntatore o un array. Di conseguenza abbiamo sfruttato puntatori, ossia una variabile che contiene l'indirizzo di memoria di un'altra variabile.
Altra funzione che ha semplificato il lavoro è lo *switch* che ci permette di assegnare diverse istruzioni da eseguire in base al valore che una variabile contiene. Per eseguire confronti abbiamo anche sfruttato le funzioni *if* e *else* le quali ci permettono di controllare se una condizione viene verificata e eseguire una specifica istruzione.
Infine abbiamo sfruttato i cicli, quindi delle funzioni che eseguono un istruzione fintanto che una determinata condizione non viene verificata, di questo si occupano le funzioni *for* e *while*.
Fondamentale per la gestione degli errori è la funzione *exit* che chiude tutti i processi del gioco nel caso in cui si incontrino degli errori.