Davide Ciacco 794163

Progetto Assembly

Generatore procedurale di labirinti e pathfinder

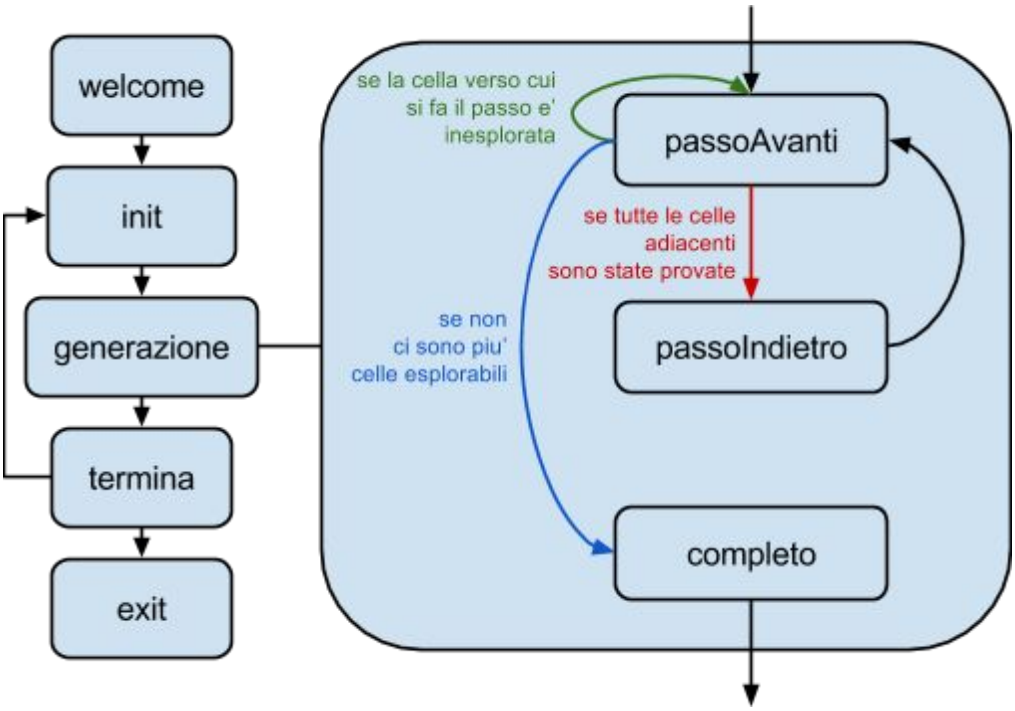
Questo programma è un generatore procedurale di labirinti (2D).
Accetta in input un seed per inizializzare la funzione di generazione di numeri pseudo-casuali e le dimensioni (larghezza e altezza).
Le dimensioni accettabili sono limitate tra 2 e 16 compresi, larghezza e altezza possono essere diverse tra loro.

Dando seed, larghezza e altezza uguali si otterrà sempre lo stesso labirinto.

L'algoritmo che ho usato e' una versione procedurale del recursive backtracker in cui al posto di scegliere la prima posizione libera se ne sceglie una, a caso, tra quelle libere.
Recursive backtracker:
1) se tutte le celle del labirinto sono state esplorate, vai a 4
2) se non ci sono altre celle adiacenti esplorabili dalla posizione in cui ci si trova, torna alla cella precedente
3) fai un passo avanti in una cella adiacente inesplorata, torna a 1
4) termina

A livello logico il labirinto e' considerabile come una matrice a due dimensioni (larghezza*altezza) in cui ogni cella puo' essere individuata da delle coordinate X,Y.
A livello pratico e' una stringa unidimensionale di $(2 \cdot \text{larghezza}+2) \cdot (2 \cdot \text{altezza}+1)+1$ caratteri.
Date le coordinate X e Y e' possibile trovare il punto (p) corrispondente sulla stringa usando questa equazione:
$p=(2 \cdot X)+(4 \cdot \text{larghezza}+4) \cdot Y+(2 \cdot \text{larghezza}+3)$

Struttura del codice:



init:
- chiede all'utente le dimensioni del labirinto
- scrive in memoria la stringa che sara' utilizzata per rappresentare il labirinto
- chiede all'utente di inserire il seed necessario per generare numeri pseudocasuali
- genera proceduralmente le coordinate di partenza
- calcola la posizione delle coordinate sulla stringa del labirinto

generazione:
- la funzione di generazione e' formata da 3 parti: *passoAvanti*, *passoIndietro* e *completo*

passoAvanti:
- se tutte le celle sono state esplorate, passa a *completo*
- se tutte le direzioni dalla cella in cui ci si trova sono state provate, passa a *passoIndietro*
- determina in che direzione provera' a muoversi
- se puo' muoversi in quella direzione salva nella stack lo stato di esplorazione per quella cella e torna a *passoAvanti* (si trovera' pero' in una nuova cella)
- se non puo' muoversi in quella direzione riprova con un'altra

passoIndietro:
- torna indietro verso la direzione di provenienza, cancellando il passo fatto precedentemente
- recupera i dati dalla stack relativi alla cella in cui e' tornato

completo:
- continua a tornare indietro (deallocando la stack) fino a quando non si ritrova al punto di partenza
- conta i passi necessari per arrivare da A a B
- la generazione del labirinto e' completa, passa a termina

termina:
- chiede all'utente se vuole terminare il programma o se vuole generare un nuovo labirinto
- se vuole generare un nuovo labirinto, passa a init
- se vuole uscire, stampa un messaggio di saluto e termina il programma

Altre procedure implementate:
- seed: chiede all'utente di inserire un seed per inizializzare il generatore di numeri pseudocasuali, controlla che il seed sia diverso da 0
- rand: implementazione del generatore di numeri pseudocasuali, usa il metodo XORSHIFT con uno shift a destra di 3 e a sinistra di 5
- storeChar: salva un carattere(byte) sulla stringa del labirinto

Avrei potuto usare l'istruzione *srbk* per allocare memoria dinamicamente, ma ho preferito allocare manualmente la dimensione massima in bytes che la stringa di un labirinto 16x16 (il più grande concesso) potrebbe occupare.

Editor di testo utilizzati: Atom e SublimeText
Il programma è stato testato sia su qtSpim che su MARS (su Windows, Linux e MacOS)