Snake Labyrinth

Generated by Doxygen 1.9.1

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 3

Class Documentation

3.1 labyrinth_player Struct Reference

Contiene tutti i valori attuali del giocatore.

`#include <game_manager.h>`

Public Attributes

- int position [2]

  position[0] è la riga attuale, position[1] è la colonna attuale
- bool won

  flag per vedere se il giocatore ha vinto
- char * moves_storage

  contiene tutte le mosse eseguite
- int moves_counter

  contiene il numero di mosse eseguite
- int score

  tiene conto del punteggio
- char **left**
- char **up**
- char **down**
- char right

  contiene il simbolo usato per andare nella rispettiva direzione
- int drill

  tiene conto di quanti muri si possono attraversare

3.1.1 Detailed Description

Contiene tutti i valori attuali del giocatore.

The documentation for this struct was generated from the following file:

- game_manager.h

3.2 labyrinth_stage Struct Reference

Contiene la matrice su cui si svolge il gioco e la sua grandezza.

`#include <game_manager.h>`

Public Attributes

- char ** playground
  contiene il livello
- int rows
  righe del labirinto
- int columns
  colonne del labirinto

3.2.1 Detailed Description

Contiene la matrice su cui si svolge il gioco e la sua grandezza.

The documentation for this struct was generated from the following file:

- game_manager.h

3.3 tail Struct Reference

Linked list che contiene le coordinate di ogni elemento della coda di snake.

`#include <tail.h>`

Collaboration diagram for tail:

Public Attributes

- int rows
  riga dell'elemento della coda
- int columns
  colonna dell'elemento della coda
- struct tail * next
  puntatore al prossimo elemento della coda

3.3.1 Detailed Description

Linked list che contiene le coordinate di ogni elemento della coda di snake.

The documentation for this struct was generated from the following file:

- tail.h

Chapter 4

File Documentation

4.1 challenges.h File Reference

Contiene l'algoritmo usato per la soluzione delle challenge.

This graph shows which files directly or indirectly include this file:

4.2 game_manager.h File Reference

Logica del gioco.

This graph shows which files directly or indirectly include this file:

Classes

- struct labyrinth_stage

  Contiene la matrice su cui si svolge il gioco e la sua grandezza.
- struct labyrinth_player

  Contiene tutti i valori attuali del giocatore.

Functions

- void show_tail (list ∗tail, labyrinth_stage ∗stage)

  Per ogni elemento della coda, mette il simbolo x nella matrice di gioco.
- void delete_old_tail (labyrinth_stage ∗stage)

  Cancella la coda (all'occhio dell'utente) dalla matrice.
- void show_stages (bool loaded, list ∗tail, labyrinth_stage ∗stage)

  Stampa la matrice di gioco.
- void load_game (char ∗stage_no, int method, labyrinth_stage ∗stage, labyrinth_player ∗player)

  Carica in memoria la matrice di gioco e inizializza i valori della struct labyrinth_player.
- void store_move (char direction, labyrinth_player ∗player)

  Inserisce dentro a moves_storage la mossa passata come parametro.
- void free_game (labyrinth_stage ∗stage, labyrinth_player ∗player, list ∗tail)

  Dealloca dalla memoria la matrice di gioco e gli altri dati precedentemenenti allocati prima di finire l'esecuzione del programma.
- void clear ()

  Pulisco l'interfaccia del terminale.

4.2.1 Detailed Description

Logica del gioco.

4.2.2 Function Documentation

4.2.2.1 clear()

```
void clear ( )
```

Pulisco l'interfaccia del terminale.

se l'OS è unix like, verrà usato il comando clear, altrimenti viene usato cls per windows

4.2.2.2 load_game()

```
void load_game (
            char * stage_no,
            int method,
            labyrinth_stage * stage,
            labyrinth_player * player )
```

Carica in memoria la matrice di gioco e inizializza i valori della struct labyrinth_player.

la matrice può essere caricata in due modi: inserendola manualmente oppure caricandola dal file contenente alcuni livelli già fatti

4.2.2.3 show_stages()

```
void show_stages (
            bool loaded,
            list * tail,
            labyrinth_stage * stage )
```

Stampa la matrice di gioco.

se il livello deve ancora essere selezionato stampo tutto il file contenente i livelli, altrimenti stampo il livello caricato in memoria

4.3 main.c File Reference

File principale.

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
#include <time.h>
#include "tail.h"
#include "game_manager.h"
#include "movement.h"
#include "challenges.h"
```

Include dependency graph for main.c:

Functions

- int main (int argc, char **argv)

4.3.1 Detailed Description

File principale.

4.3.2 Function Documentation

4.3.2.1 main()

```
int main (
            int argc,
            char ** argv )
```

1. Se non è presente –challegne come parametro, faccio scegliere la modalità interattiva/random all'utente
2. Viene chiesto come si vuole selezionare il livello: input da tastiera specificando righe e colonne del livello, oppure un livello già esistente sul file labyrinth.txt
3. Viene caricata la matrice in memoria e si comincia a giocare
4. Dopo ogni mossa l'interfaccia del terminale viene pulita
5. Alla fine viene chiesto se si vogliono vedere le mosse eseguite (potrebbero essere tante)
6. Viene deallocato dalla memoria quello che è stato allocato precedentemente (matrice, coda, storage delle mosse eseguite)

4.4 movement.h File Reference

Movimento di snake.

This graph shows which files directly or indirectly include this file:

Functions

- int validate_move (char next_position, int *next_coordinates, list **tail, labyrinth_player *player)

  Gestisce i vari casi che si possono verificare prima di muovere effettivamente snake.
- void move (char direction, list **tail, labyrinth_stage *stage, labyrinth_player *player)

  Muove la testa di snake.
- void move_tail (list **tail, int rows, int columns)

  Muove la coda di snake.

4.4.1 Detailed Description

Movimento di snake.

4.4.2 Function Documentation

4.4.2.1 move()

```
void move (
            char direction,
            list ** tail,
            labyrinth_stage * stage,
            labyrinth_player * player )
```

Muove la testa di snake.

Se non ha ancora una coda, lascierà dietro di sè il simbolo '.' inoltre prima di eseguire una mossa verifica se è possibile eseguirla

4.4.2.2 validate_move()

```
int validate_move (
            char next_position,
            int * next_coordinates,
            list ** tail,
            labyrinth_player * player )
```

Gestisce i vari casi che si possono verificare prima di muovere effettivamente snake.

Se trovo:
$: aggiungo un elemento alla coda e aggiungo 10 punti al giocatore
_: il giocatore ha vinto
!: i punti vengono dimezzati, lo stesso per la coda
x: trovo quale è l'indice di quell'elemento della coda ed eseguo n volte la funzione pop in base al valore dell'indice
T: il giocatore può attraversare altre 3 pareti (non può andare fuori dal labirinto)
#: avendo il trapano posso superare la parete, inoltre diminuisco gli usi possibili del trapano di 1

4.5 tail.h File Reference

Gestione della coda di snake.

This graph shows which files directly or indirectly include this file:

Classes

- struct tail

  Linked list che contiene le coordinate di ogni elemento della coda di snake.

Typedefs

- typedef struct tail list

  Linked list che contiene le coordinate di ogni elemento della coda di snake.

Functions

- list ∗ l_create (int rows, int columns)

  Crea un nuovo elemento nella coda.
- void list_append (list ∗∗l_orig, int rows, int columns)

  Aggiunge un nuovo elemento alla coda.
- void pop (list ∗∗l_orig)

  Rimuove l'ultimo elemento dalla coda.
- void print_list (list ∗l)

  Stampa i contenuti della linked list.
- int get_tail_length (list ∗l)

  Ottiene quanti elementi sono presenti nella linked list.
- int get_node_index (list ∗l, int rows, int columns)

  Ritorna la posizione dell'elemento della lista che possiede i valori passati come parametri alla funzione.
- void delete_tail (list ∗l)

  Libera la memoria allocata dalla linked list, viene chiamata da free_game prima di finire il programma.

4.5.1 Detailed Description

Gestione della coda di snake.

Index