CDungeon

1.0

Generato da Doxygen 1.16.1

1 Gerarchia delle directory 1

1.1 Directory . . . 1

2 Indice delle strutture dati 3

2.1 Strutture dati . . . 3

3 Indice dei file 5

3.1 Elenco dei file . . . 5

4 Documentazione delle directory 7

4.1 Riferimenti per la directory src . . . 7

5 Documentazione delle classi 9

5.1 Riferimenti per la struct Giocatore . . . 9

5.1.1 Descrizione dettagliata . . . 9

5.1.2 Documentazione dei campi . . . 9

5.1.2.1 ha_armatura . . . 9

5.1.2.2 ha_chiave_castello . . . 10

5.1.2.3 ha_spada . . . 10

5.1.2.4 ha_spada_eroe . . . 10

5.1.2.5 max_punti_vita . . . 10

5.1.2.6 missione_grotta . . . 10

5.1.2.7 missione_magione . . . 11

5.1.2.8 missione_palude . . . 11

5.1.2.9 monete . . . 11

5.1.2.10 numero_oggetti . . . 11

5.1.2.11 punti_vita . . . 11

5.2 Riferimenti per la struct NodoSalvataggio . . . 12

5.2.1 Descrizione dettagliata . . . 12

5.2.2 Documentazione dei campi . . . 12

5.2.2.1 dati_giocatore . . . 12

5.2.2.2 id . . . 12

5.2.2.3 prossimo . . . 12

5.2.2.4 timestamp . . . 13

5.3 Riferimenti per la struct Stanza . . . 13

5.3.1 Descrizione dettagliata . . . 13

5.3.2 Documentazione dei campi . . . 13

5.3.2.1 colpo_fatale . . . 13

5.3.2.2 danno . . . 14

5.3.2.3 is_boss . . . 14

5.3.2.4 nome . . . 14

5.3.2.5 ricompensa_monete . . . 14

5.3.2.6 tipo . . . 14

6 Documentazione dei file **15**

6.1 Riferimenti per il file src/combattimento.c . . . 15
6.1.1 Descrizione dettagliata . . . 15
6.1.2 Documentazione delle funzioni . . . 15
6.1.2.1 applica_danno_trappola() . . . 15
6.1.2.2 combattimento_boss_finale() . . . 16
6.1.2.3 inizia_combattimento() . . . 16
6.2 combattimento.c . . . 17
6.3 Riferimenti per il file src/combattimento.h . . . 18
6.3.1 Descrizione dettagliata . . . 19
6.3.2 Documentazione delle funzioni . . . 19
6.3.2.1 applica_danno_trappola() . . . 19
6.3.2.2 combattimento_boss_finale() . . . 19
6.3.2.3 inizia_combattimento() . . . 20
6.4 combattimento.h . . . 20
6.5 Riferimenti per il file src/dungeon.c . . . 21
6.5.1 Descrizione dettagliata . . . 21
6.5.2 Documentazione delle funzioni . . . 21
6.5.2.1 esegui_missione() . . . 21
6.5.2.2 genera_stanza_missione() . . . 22
6.6 dungeon.c . . . 22
6.7 Riferimenti per il file src/dungeon.h . . . 26
6.7.1 Descrizione dettagliata . . . 26
6.7.2 Documentazione delle funzioni . . . 27
6.7.2.1 esegui_missione() . . . 27
6.7.2.2 genera_stanza_missione() . . . 27
6.8 dungeon.h . . . 28
6.9 Riferimenti per il file src/giocatore.c . . . 28
6.9.1 Descrizione dettagliata . . . 28
6.9.2 Documentazione delle funzioni . . . 28
6.9.2.1 inizializza_giocatore() . . . 28
6.9.2.2 stampa_statistiche_giocatore() . . . 29
6.10 giocatore.c . . . 29
6.11 Riferimenti per il file src/giocatore.h . . . 30
6.11.1 Descrizione dettagliata . . . 30
6.11.2 Documentazione delle funzioni . . . 30
6.11.2.1 inizializza_giocatore() . . . 30
6.11.2.2 stampa_statistiche_giocatore() . . . 31
6.12 giocatore.h . . . 31
6.13 Riferimenti per il file src/main.c . . . 32
6.13.1 Descrizione dettagliata . . . 32
6.13.2 Documentazione delle funzioni . . . 32

6.13.2.1 main() . . . 32
6.14 main.c . . . 33
6.15 Riferimenti per il file src/menu.c . . . 33
6.15.1 Descrizione dettagliata . . . 34
6.15.2 Documentazione delle funzioni . . . 34
6.15.2.1 gestisci_trucchi() . . . 34
6.15.2.2 mostra_menu_missione() . . . 34
6.15.2.3 mostra_menu_principale() . . . 34
6.15.2.4 mostra_menu_salvataggi() . . . 35
6.15.2.5 mostra_menu_villaggio() . . . 35
6.15.2.6 mostra_negozio() . . . 36
6.16 menu.c . . . 36
6.17 Riferimenti per il file src/menu.h . . . 40
6.17.1 Descrizione dettagliata . . . 41
6.17.2 Documentazione delle funzioni . . . 41
6.17.2.1 gestisci_trucchi() . . . 41
6.17.2.2 mostra_menu_missione() . . . 41
6.17.2.3 mostra_menu_principale() . . . 42
6.17.2.4 mostra_menu_salvataggi() . . . 42
6.17.2.5 mostra_menu_villaggio() . . . 43
6.17.2.6 mostra_negozio() . . . 43
6.18 menu.h . . . 43
6.19 Riferimenti per il file src/salvataggio.c . . . 44
6.19.1 Descrizione dettagliata . . . 44
6.19.2 Documentazione delle funzioni . . . 44
6.19.2.1 aggiungi_salvataggio() . . . 44
6.19.2.2 carica_salvataggi_da_file() . . . 45
6.19.2.3 carica_salvataggio() . . . 45
6.19.2.4 elimina_salvataggio() . . . 46
6.19.2.5 libera_salvataggi() . . . 46
6.19.2.6 ottieni_data_corrente() . . . 46
6.19.2.7 salva_tutto_su_file() . . . 46
6.19.2.8 stampa_salvataggi() . . . 47
6.20 salvataggio.c . . . 47
6.21 Riferimenti per il file src/salvataggio.h . . . 50
6.21.1 Descrizione dettagliata . . . 50
6.21.2 Documentazione delle ridefinizioni di tipo (typedef) . . . 51
6.21.2.1 NodoSalvataggio . . . 51
6.21.3 Documentazione delle funzioni . . . 51
6.21.3.1 aggiungi_salvataggio() . . . 51
6.21.3.2 carica_salvataggi_da_file() . . . 51
6.21.3.3 carica_salvataggio() . . . 51

6.21.3.4 elimina_salvataggio() . . . 52
6.21.3.5 libera_salvataggi() . . . 52
6.21.3.6 salva_tutto_su_file() . . . 53
6.21.3.7 stampa_salvataggi() . . . 53
6.22 salvataggio.h . . . 54
6.23 Riferimenti per il file src/stanza.h . . . 54
6.23.1 Descrizione dettagliata . . . 54
6.23.2 Documentazione dei tipi enumerati . . . 54
6.23.2.1 TipoStanza . . . 54
6.24 stanza.h . . . 55
6.25 Riferimenti per il file src/utilita.c . . . 55
6.25.1 Descrizione dettagliata . . . 56
6.25.2 Documentazione delle funzioni . . . 56
6.25.2.1 controlla_padovan() . . . 56
6.25.2.2 lancia_dado() . . . 56
6.25.2.3 leggi_input_char() . . . 57
6.25.2.4 leggi_intero() . . . 57
6.25.2.5 padovan() . . . 57
6.25.2.6 pulisci_schermo() . . . 58
6.25.2.7 valuta_vittoria_morra() . . . 58
6.26 utilita.c . . . 59
6.27 Riferimenti per il file src/utilita.h . . . 60
6.27.1 Descrizione dettagliata . . . 60
6.27.2 Documentazione delle funzioni . . . 60
6.27.2.1 controlla_padovan() . . . 60
6.27.2.2 lancia_dado() . . . 61
6.27.2.3 leggi_input_char() . . . 61
6.27.2.4 leggi_intero() . . . 62
6.27.2.5 padovan() . . . 62
6.27.2.6 pulisci_schermo() . . . 62
6.27.2.7 valuta_vittoria_morra() . . . 63
6.28 utilita.h . . . 63

Chapter 1

Gerarchia delle directory

1.1 Directory

src 7
combattimento.c 15
combattimento.h 18
dungeon.c 21
dungeon.h 26
giocatore.c 28
giocatore.h 30
main.c 32
menu.c 33
menu.h 40
salvataggio.c 44
salvataggio.h 50
stanza.h 54
utilita.c 55
utilita.h 60

Chapter 2

Indice delle strutture dati

2.1 Strutture dati

Queste sono le strutture dati con una loro breve descrizione:

Giocatore
: Struttura che rappresenta lo stato del giocatore 9

NodoSalvataggio
: Nodo di una lista concatenata per memorizzare i salvataggi 12

Stanza
: Struttura che rappresenta una stanza del dungeon 13

Chapter 3

Indice dei file

3.1 Elenco dei file

Questo è un elenco di tutti i file con una loro breve descrizione:

src/combattimento.c
: Implementazione delle logiche di combattimento . . . 15

src/combattimento.h
: Gestione del combattimento e delle interazioni con nemici, trappole e boss . . . 18

src/dungeon.c
: Implementazione della logica del dungeon e delle missioni . . . 21

src/dungeon.h
: Gestione della generazione e del flusso del dungeon . . . 26

src/giocatore.c
: Implementazione delle funzioni relative al giocatore . . . 28

src/giocatore.h
: Definizione della struttura Giocatore e funzioni correlate . . . 30

src/main.c
: Entry point del gioco CDungeon . . . 32

src/menu.c
: Implementazione dei menu e dell'interfaccia utente . . . 33

src/menu.h
: Gestione dei menu e dell'interfaccia utente . . . 40

src/salvataggio.c
: Implementazione delle funzioni per la gestione dei salvataggi . . . 44

src/salvataggio.h
: Gestione del sistema di salvataggio e caricamento . . . 50

src/stanza.h
: Definizione della struttura e dei tipi di Stanza . . . 54

src/utilita.c
: Implementazione delle funzioni di utilità . . . 55

src/utilita.h
: Funzioni di utilità generale per il gioco . . . 60

Chapter 4

Documentazione delle directory

4.1 Riferimenti per la directory src

File

- file combattimento.c

  Implementazione delle logiche di combattimento.
- file combattimento.h

  Gestione del combattimento e delle interazioni con nemici, trappole e boss.
- file dungeon.c

  Implementazione della logica del dungeon e delle missioni.
- file dungeon.h

  Gestione della generazione e del flusso del dungeon.
- file giocatore.c

  Implementazione delle funzioni relative al giocatore.
- file giocatore.h

  Definizione della struttura Giocatore e funzioni correlate.
- file main.c

  Entry point del gioco CDungeon.
- file menu.c

  Implementazione dei menu e dell'interfaccia utente.
- file menu.h

  Gestione dei menu e dell'interfaccia utente.
- file salvataggio.c

  Implementazione delle funzioni per la gestione dei salvataggi.
- file salvataggio.h

  Gestione del sistema di salvataggio e caricamento.
- file stanza.h

  Definizione della struttura e dei tipi di Stanza.
- file utilita.c

  Implementazione delle funzioni di utilità.
- file utilita.h

  Funzioni di utilità generale per il gioco.

Chapter 5

Documentazione delle classi

5.1 Riferimenti per la struct Giocatore

Struttura che rappresenta lo stato del giocatore.

`#include <giocatore.h>`

Campi

- int punti_vita
- int max_punti_vita
- int monete
- int numero_oggetti
- int missione_palude
- int missione_magione
- int missione_grotta
- int ha_spada
- int ha_armatura
- int ha_spada_eroe
- int ha_chiave_castello

5.1.1 Descrizione dettagliata

Struttura che rappresenta lo stato del giocatore.

Definizione alla linea 12 del file giocatore.h.

5.1.2 Documentazione dei campi

5.1.2.1 ha_armatura

`int Giocatore::ha_armatura`

1 se possiede l'armatura, 0 altrimenti

Definizione alla linea 26 del file giocatore.h.

Referenziato da applica_danno_trappola(), inizia_combattimento(), inizializza_giocatore(), mostra_negozio(), e stampa_statistiche_giocatore().

5.1.2.2 ha_chiave_castello

```
int Giocatore::ha_chiave_castello
```

1 se possiede la chiave del castello, 0 altrimenti

Definizione alla linea 28 del file giocatore.h.

Referenziato da esegui_missione(), inizializza_giocatore(), e stampa_statistiche_giocatore().

5.1.2.3 ha_spada

```
int Giocatore::ha_spada
```

1 se possiede la spada base, 0 altrimenti

Definizione alla linea 25 del file giocatore.h.

Referenziato da inizia_combattimento(), inizializza_giocatore(), mostra_negozio(), e stampa_statistiche_giocatore().

5.1.2.4 ha_spada_eroe

```
int Giocatore::ha_spada_eroe
```

1 se possiede la spada dell'eroe, 0 altrimenti

Definizione alla linea 27 del file giocatore.h.

Referenziato da esegui_missione(), inizia_combattimento(), inizializza_giocatore(), e stampa_statistiche_giocatore().

5.1.2.5 max_punti_vita

```
int Giocatore::max_punti_vita
```

Punti vita massimi del giocatore

Definizione alla linea 15 del file giocatore.h.

Referenziato da gestisci_trucchi(), inizializza_giocatore(), mostra_menu_villaggio(), mostra_negozio(), e stampa_statistiche_giocatore().

5.1.2.6 missione_grotta

```
int Giocatore::missione_grotta
```

1 se la missione grotta è completata, 0 altrimenti

Definizione alla linea 22 del file giocatore.h.

Referenziato da gestisci_trucchi(), inizializza_giocatore(), mostra_menu_missione(), stampa_salvataggi(), e stampa_statistiche_giocatore().

5.1.2.7 missione_magione

```
int Giocatore::missione_magione
```

1 se la missione magione è completata, 0 altrimenti

Definizione alla linea 21 del file giocatore.h.

Referenziato da gestisci_trucchi(), inizializza_giocatore(), mostra_menu_missione(), stampa_salvataggi(), e stampa_statistiche_giocatore().

5.1.2.8 missione_palude

```
int Giocatore::missione_palude
```

1 se la missione palude è completata, 0 altrimenti

Definizione alla linea 20 del file giocatore.h.

Referenziato da gestisci_trucchi(), inizializza_giocatore(), mostra_menu_missione(), stampa_salvataggi(), e stampa_statistiche_giocatore().

5.1.2.9 monete

```
int Giocatore::monete
```

Numero di monete possedute

Definizione alla linea 16 del file giocatore.h.

Referenziato da esegui_missione(), gestisci_trucchi(), inizia_combattimento(), inizializza_giocatore(), mostra_menu_villaggio(), mostra_negozio(), stampa_salvataggi(), e stampa_statistiche_giocatore().

5.1.2.10 numero_oggetti

```
int Giocatore::numero_oggetti
```

Conteggio generico oggetti

Definizione alla linea 17 del file giocatore.h.

Referenziato da esegui_missione(), inizializza_giocatore(), mostra_negozio(), stampa_salvataggi(), e stampa_statistiche_giocatore().

5.1.2.11 punti_vita

```
int Giocatore::punti_vita
```

Punti vita attuali del giocatore

Definizione alla linea 14 del file giocatore.h.

Referenziato da applica_danno_trappola(), esegui_missione(), gestisci_trucchi(), inizia_combattimento(), inizializza_giocatore(), mostra_menu_missione(), mostra_menu_villaggio(), mostra_negozio(), stampa_salvataggi(), e stampa_statistiche_giocatore().

La documentazione per questa struct è stata generata a partire dal seguente file:

- src/giocatore.h

5.2 Riferimenti per la struct NodoSalvataggio

Nodo di una lista concatenata per memorizzare i salvataggi.

`#include <salvataggio.h>`

Diagramma di collaborazione per NodoSalvataggio:

Campi

- int id
- char timestamp [32]
- Giocatore dati_giocatore
- struct NodoSalvataggio * prossimo

5.2.1 Descrizione dettagliata

Nodo di una lista concatenata per memorizzare i salvataggi.

Definizione alla linea 14 del file salvataggio.h.

5.2.2 Documentazione dei campi

5.2.2.1 dati_giocatore

`Giocatore NodoSalvataggio::dati_giocatore`

Copia dello stato del giocatore

Definizione alla linea 18 del file salvataggio.h.

Referenziato da aggiungi_salvataggio(), carica_salvataggi_da_file(), gestisci_trucchi(), mostra_menu_salvataggi(), salva_tutto_su_file(), e stampa_salvataggi().

5.2.2.2 id

`int NodoSalvataggio::id`

Identificativo univoco del salvataggio

Definizione alla linea 16 del file salvataggio.h.

Referenziato da aggiungi_salvataggio(), carica_salvataggi_da_file(), carica_salvataggio(), elimina_salvataggio(), salva_tutto_su_file(), e stampa_salvataggi().

5.2.2.3 prossimo

```
struct NodoSalvataggio* NodoSalvataggio::prossimo
```

Puntatore al prossimo nodo della lista

Definizione alla linea 19 del file salvataggio.h.

Referenziato da aggiungi_salvataggio(), carica_salvataggi_da_file(), carica_salvataggio(), elimina_salvataggio(), libera_salvataggi(), salva_tutto_su_file(), e stampa_salvataggi().

5.2.2.4 timestamp

```
char NodoSalvataggio::timestamp[32]
```

Data e ora di creazione del salvataggio

Definizione alla linea 17 del file salvataggio.h.

Referenziato da aggiungi_salvataggio(), carica_salvataggi_da_file(), salva_tutto_su_file(), e stampa_salvataggi().

La documentazione per questa struct è stata generata a partire dal seguente file:

- src/salvataggio.h

5.3 Riferimenti per la struct Stanza

Struttura che rappresenta una stanza del dungeon.

```
#include <stanza.h>
```

Campi

- char nome [32]
- TipoStanza tipo
- int danno
- int colpo_fatale
- int ricompensa_monete
- int is_boss

5.3.1 Descrizione dettagliata

Struttura che rappresenta una stanza del dungeon.

Definizione alla linea 23 del file stanza.h.

5.3.2 Documentazione dei campi

5.3.2.1 colpo_fatale

`int Stanza::colpo_fatale`

Danno che il giocatore deve infliggere per sconfiggere il nemico

Definizione alla linea 28 del file stanza.h.

Referenziato da genera_stanza_missione(), e inizia_combattimento().

5.3.2.2 danno

`int Stanza::danno`

Danno inflitto dalla stanza al giocatore

Definizione alla linea 27 del file stanza.h.

Referenziato da applica_danno_trappola(), esegui_missione(), genera_stanza_missione(), e inizia_combattimento().

5.3.2.3 is_boss

`int Stanza::is_boss`

Indica se la stanza contiene un boss

Definizione alla linea 30 del file stanza.h.

Referenziato da esegui_missione(), e genera_stanza_missione().

5.3.2.4 nome

`char Stanza::nome[32]`

Nome della stanza

Definizione alla linea 25 del file stanza.h.

Referenziato da applica_danno_trappola(), esegui_missione(), genera_stanza_missione(), e inizia_combattimento().

5.3.2.5 ricompensa_monete

`int Stanza::ricompensa_monete`

Ricompensa in monete per il giocatore

Definizione alla linea 29 del file stanza.h.

Referenziato da esegui_missione(), genera_stanza_missione(), e inizia_combattimento().

5.3.2.6 tipo

`TipoStanza Stanza::tipo`

Tipo di stanza (nemico, trappola, vuota)

Definizione alla linea 26 del file stanza.h.

Referenziato da esegui_missione(), e genera_stanza_missione().

La documentazione per questa struct è stata generata a partire dal seguente file:

- src/stanza.h

Chapter 6

Documentazione dei file

6.1 Riferimenti per il file src/combattimento.c

Implementazione delle logiche di combattimento.

```
#include "combattimento.h"
#include "utilita.h"
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <stdlib.h>
```

Grafo delle dipendenze di inclusione per combattimento.c:

6.2 combattimento.c

Vai alla documentazione di questo file.

```
00001
00005
00006 #include "combattimento.h"
00007 #include "utilita.h"
00008 #include <stdio.h>
00009 #include <string.h>
00010 #include <ctype.h>
00011 #include <stdlib.h>
00012
00013 int inizia_combattimento(Giocatore *g, Stanza *e)
00014 {
00015   printf("\n--- COMBATTIMENTO: %s ---\n", e->nome);
00016
00017   while (g->punti_vita > 0)
00018   {
00019     printf("\nTu: %d HP | Nemico: %s\n", g->punti_vita, e->nome);
00020     printf("Premi Invio per attaccare...");
00021     getchar();
00022
00023     int danno_nemico = e->danno;
00024
00025     // Controllo colpo fatale Generale Orco
00026     if (strcmp(e->nome, "Generale Orco") == 0 && g->ha_spada_eroe)
00027     {
00028       e->colpo_fatale = 5;
00029     }
00030
00031     // Controllo indovinello Drago
00032     if (strcmp(e->nome, "Drago Antico") == 0)
00033     {
00034       int n = (rand() % 500) + 1;
00035       printf("Drago: Il numero %d e' di Padovan? [s/n]: ", n);
```

```
00036       char r;
00037       scanf(" %c", &r);
00038       getchar();
00039       int vero = controlla_padovan(n);
00040       if ((tolower(r) == 's' && vero) || (tolower(r) == 'n' && !vero))
00041       {
00042         printf("Giusto! Il Drago non attacca.\n");
00043         danno_nemico = 0;
00044       }
00045       else
00046         printf("Sbagliato! Grrr!\n");
00047     }
00048
00049     // Turno Giocatore
00050     int dado = lancia_dado(6);
00051     int attacco = dado;
00052
00053     if (g->ha_spada_eroe)
00054       attacco += 2;
00055     else if (g->ha_spada)
00056       attacco += 1;
00057
00058     printf("Tiro: %d (Totale: %d)\n", dado, attacco);
00059
00060     if (attacco > e->colpo_fatale)
00061     {
00062       printf("Colpo fatale! %s sconfitto!\n", e->nome);
00063       g->monete += e->ricompensa_monete;
00064       printf("Ottieni %d monete.\n", e->ricompensa_monete);
00065       return 1;
00066     }
00067     else
00068     {
00069       printf("Mancato (Serviva > %d).\n", e->colpo_fatale);
00070
00071       // Turno Nemico
00072       if (g->ha_armatura)
00073       {
00074         danno_nemico--;
00075         if(danno_nemico < 0)
00076           danno_nemico = 0;
00077         printf("Armatura assorbe 1 danno.\n");
00078       }
00079       g->punti_vita -= danno_nemico;
00080       printf("%s infligge %d danni!\n", e->nome, danno_nemico);
00081     }
00082   }
00083
00084   printf("\nSEI STATO SCONFITTO!\n");
00085   g->punti_vita = 0;
00086   getchar();
00087   return 0;
00088 }
00089
00090 void applica_danno_trappola(Giocatore *g, Stanza *trappola)
00091 {
00092   printf("\n!!! TRAPPOLA: %s !!!\n", trappola->nome);
00093
00094   int danno_trappola = trappola->danno;
00095
00096   if (g->ha_armatura)
00097   {
00098     danno_trappola--;
00099     if (danno_trappola < 0)
00100       danno_trappola = 0;
00101     printf("Armatura assorbe 1 danno.\n");
00102   }
00103
00104   g->punti_vita -= danno_trappola;
00105   printf("Subisci %d danni! (HP: %d)\n", danno_trappola, g->punti_vita);
00106 }
00107
00108 int combattimento_boss_finale(Giocatore *g)
00109 {
00110   printf("\n--- SCONTRO FINALE: SIGNORE OSCURO ---\n");
00111   printf("Regole: Scudo > Spada > Magia > Scudo\n");
00112   printf("Vinci 3 round su 5.\n");
00113
00114   int vittorie_eroe = 0;
00115   int vittorie_boss = 0;
00116   int round = 1;
00117   const char *mosse[] = {"", "Scudo", "Magia", "Spada"};
00118
00119   while (vittorie_eroe < 3 && vittorie_boss < 3 && round <= 5)
00120   {
00121     printf("\n--- Round %d (Eroe %d - Boss %d) ---\n", round, vittorie_eroe,
00122            vittorie_boss);
```

```
00123     printf("1. Scudo\n2. Magia\n3. Spada\nScelta: ");
00124
00125     // Turno Giocatore
00126     int mossa_eroe = leggi_intero();
00127     if (mossa_eroe < 1 || mossa_eroe > 3)
00128       continue;
00129
00130     // Turno Boss
00131     int mossa_boss = lancia_dado(3);
00132     printf("Tu: %s | Boss: %s\n", mosse[mossa_eroe], mosse[mossa_boss]);
00133
00134     // Valutazione esito
00135     int esito = valuta_vittoria_morra(mossa_eroe, mossa_boss);
00136
00137     if (esito == 0)
00138     {
00139       printf("Pareggio!\n");
00140     }
00141     else if (esito == 1)
00142     {
00143       printf("Vinci il round!\n");
00144       vittorie_eroe++;
00145       round++;
00146     }
00147     else
00148     {
00149       printf("Il Boss vince il round!\n");
00150       vittorie_boss++;
00151       round++;
00152     }
00153   }
00154
00155   return (vittorie_eroe >= 3);
00156 }
```

6.3 Riferimenti per il file src/combattimento.h

Gestione del combattimento e delle interazioni con nemici, trappole e boss.

```
#include "stanza.h"
#include "giocatore.h"
```

Grafo delle dipendenze di inclusione per combattimento.h: Questo grafo mostra quali altri file includono direttamente o indirettamente questo file:

Funzioni

- int inizia_combattimento (Giocatore *g, Stanza *nemico)

  Inizia un combattimento tra il giocatore e un nemico.
- void applica_danno_trappola (Giocatore *g, Stanza *trappola)

  Applica i danni di una trappola al giocatore.
- int combattimento_boss_finale (Giocatore *g)

  Gestisce il boss fight finale contro il Signore Oscuro.

6.3.1 Descrizione dettagliata

Gestione del combattimento e delle interazioni con nemici, trappole e boss.

Definizione nel file combattimento.h.

6.3.2 Documentazione delle funzioni

6.3.2.1 applica_danno_trappola()

```
void applica_danno_trappola (
            Giocatore * g,
            Stanza * trappola)
```

Applica i danni di una trappola al giocatore.

Calcola i danni subiti considerando eventuali riduzioni dovute all'equipaggiamento (es. armatura).

Parametri

g	Puntatore al giocatore.
trappola	Puntatore alla stanza contenente la trappola.

Definizione alla linea 90 del file combattimento.c.

Referenzia Stanza::danno, Giocatore::ha_armatura, Stanza::nome, e Giocatore::punti_vita.

Referenziato da esegui_missione().

6.3.2.2 combattimento_boss_finale()

```
int combattimento_boss_finale (
            Giocatore * g)
```

Gestisce il boss fight finale contro il Signore Oscuro.

Implementa una versione a turni della Morra Cinese (Sasso-Carta-Forbice). Il giocatore deve vincere 3 round su 5.

Parametri

g	Puntatore al giocatore.

Restituisce

1 se il giocatore vince, 0 se perde.

Definizione alla linea 108 del file combattimento.c.

Referenzia lancia_dado(), leggi_intero(), e valuta_vittoria_morra().

Referenziato da mostra_menu_missione().

6.3.2.3 inizia_combattimento()

```
int inizia_combattimento (
            Giocatore * g,
            Stanza * nemico)
```


Inizia un combattimento tra il giocatore e un nemico.

Parametri

g	Puntatore al giocatore.
nemico	Puntatore alla stanza contenente il nemico.

Restituisce

1 se il giocatore vince, 0 se perde (Game Over).

Definizione alla linea 13 del file combattimento.c.

Referenzia Stanza::colpo_fatale, controlla_padovan(), Stanza::danno, Giocatore::ha_armatura, Giocatore::ha_spada, Giocatore::ha_spada_eroe, lancia_dado(), Giocatore::monete, Stanza::nome, Giocatore::punti_vita, e Stanza::ricompensa_monete.

Referenziato da esegui_missione().

6.4 combattimento.h

Vai alla documentazione di questo file.

```
00001
00005
00006 #ifndef COMBATTIMENTO_H
00007 #define COMBATTIMENTO_H
00008
00009 #include "stanza.h"
00010 #include "giocatore.h"
00011
00022 int inizia_combattimento(Giocatore *g, Stanza *nemico);
00023
00033 void applica_danno_trappola(Giocatore *g, Stanza *trappola);
00034
00044 int combattimento_boss_finale(Giocatore *g);
00045
00046 #endif
```

6.5 Riferimenti per il file src/dungeon.c

Implementazione della logica del dungeon e delle missioni.

```
#include "dungeon.h"
#include "combattimento.h"
#include "menu.h"
#include "utilita.h"
#include <stdio.h>
#include <string.h>
```

Grafo delle dipendenze di inclusione per dungeon.c:

Funzioni

- Stanza genera_stanza_missione (int tipo_missione, int dado)

  Genera una stanza casuale in base alla missione e al lancio del dado.
- int esegui_missione (Giocatore *g, int tipo_missione, const char *nome_missione)

  Gestisce il loop principale di esecuzione di una missione.

6.5.1 Descrizione dettagliata

Implementazione della logica del dungeon e delle missioni.

Definizione nel file dungeon.c.

6.5.2 Documentazione delle funzioni

6.5.2.1 esegui_missione()

```
int esegui_missione (
            Giocatore * g,
            int tipo_missione,
            const char * nome_missione)
```

Gestisce il loop principale di esecuzione di una missione.

Include la navigazione tra le stanze, il combattimento, il negozio e la gestione degli obiettivi della missione.

Parametri

g	Puntatore al giocatore.
tipo_missione	ID della missione da eseguire.
nome_missione	Nome visualizzato della missione.

Restituisce

1 se la missione è completata con successo, 0 in caso di fallimento o rinuncia.

Definizione alla linea 172 del file dungeon.c.

Referenzia applica_danno_trappola(), Stanza::danno, genera_stanza_missione(), Giocatore::ha_chiave_castello, Giocatore::ha_spada_eroe, inizia_combattimento(), Stanza::is_boss, lancia_dado(), leggi_intero(), Giocatore::monete, mostra_negozio(), Stanza::nome, Giocatore::numero_oggetti, pulisci_schermo(), Giocatore::punti_vita, Stanza::ricompensa_monete, stampa_statistiche_giocatore(), STANZA_NEMICO, STANZA_TRAPPOLA, e Stanza::tipo.

Referenziato da mostra_menu_missione().

6.5.2.2 genera_stanza_missione()

```
Stanza genera_stanza_missione (
            int tipo_missione,
            int dado)
```

Genera una stanza casuale in base alla missione e al lancio del dado.

Parametri

tipo_missione	ID della missione corrente (1, 2 o 3).
dado	Risultato del lancio del dado per determinare il contenuto della stanza.

Restituisce

La struct Stanza generata.

Definizione alla linea 12 del file dungeon.c.

Referenzia Stanza::colpo_fatale, Stanza::danno, Stanza::is_boss, lancia_dado(), Stanza::nome, Stanza::ricompensa_monete, STANZA_NEMICO, STANZA_NESSUNA, STANZA_TRAPPOLA, STANZA_VUOTA, e Stanza::tipo.

Referenziato da esegui_missione().

6.6 dungeon.c

Vai alla documentazione di questo file.

```
00001
00005 #include "dungeon.h"
00006 #include "combattimento.h"
00007 #include "menu.h"
00008 #include "utilita.h"
00009 #include <stdio.h>
00010 #include <string.h>
00011
00012 Stanza genera_stanza_missione(int tipo_missione, int dado)
00013 {
00014   Stanza s;
00015   s.tipo = STANZA_NESSUNA;
00016   sprintf(s.nome, "Nulla");
00017   s.danno = 0;
00018   s.colpo_fatale = 0;
00019   s.ricompensa_monete = 0;
00020   s.is_boss = 0;
00021
00022   // Configurazione entita in base a missione e dado
00023
00024   // Missione 1: Palude
00025   if (tipo_missione == 1)
00026   {
00027     if (dado == 1)
00028     {
00029       sprintf(s.nome, "Cane Selvaggio");
00030       s.tipo = STANZA_NEMICO;
00031       s.colpo_fatale = 2;
00032       s.danno = 1;
00033     }
00034     else if (dado == 2)
00035     {
00036       sprintf(s.nome, "Goblin");
00037       s.tipo = STANZA_NEMICO;
00038       s.colpo_fatale = 3;
00039       s.danno = 2;
00040       s.ricompensa_monete = 2;
00041     }
00042     else if (dado == 3)
00043     {
00044       sprintf(s.nome, "Scheletro");
00045       s.tipo = STANZA_NEMICO;
00046       s.colpo_fatale = 4;
00047       s.danno = 2;
00048       s.ricompensa_monete = 4;
00049     }
00050     else if (dado == 4)
00051     {
00052       sprintf(s.nome, "Orco");
00053       s.tipo = STANZA_NEMICO;
00054       s.colpo_fatale = 3;
```

```
00055       s.danno = 4;
00056       s.ricompensa_monete = 6;
00057     }
00058     else if (dado == 5)
00059     {
00060       sprintf(s.nome, "Acquitrino Velenoso");
00061       s.tipo = STANZA_TRAPPOLA;
00062       s.danno = lancia_dado(6);
00063     }
00064     else if (dado == 6)
00065     {
00066       sprintf(s.nome, "Generale Orco");
00067       s.tipo = STANZA_NEMICO;
00068       s.colpo_fatale = 6;
00069       s.danno = 3;
00070       s.ricompensa_monete = 12;
00071       s.is_boss = 1;
00072     }
00073   }
00074   // Missione 2: Magione
00075   else if (tipo_missione == 2)
00076   {
00077     if (dado == 1)
00078     {
00079       sprintf(s.nome, "Botola Buia");
00080       s.tipo = STANZA_TRAPPOLA;
00081       s.danno = 3;
00082     }
00083     else if (dado == 2)
00084     {
00085       sprintf(s.nome, "Pipistrello");
00086       s.tipo = STANZA_NEMICO;
00087       s.colpo_fatale = 2;
00088       s.danno = 2;
00089       s.ricompensa_monete = 1;
00090     }
00091     else if (dado == 3)
00092     {
00093       sprintf(s.nome, "Zombie");
00094       s.tipo = STANZA_NEMICO;
00095       s.colpo_fatale = 3;
00096       s.danno = 2;
00097       s.ricompensa_monete = 2;
00098     }
00099     else if (dado == 4)
00100     {
00101       sprintf(s.nome, "Fantasma");
00102       s.tipo = STANZA_NEMICO;
00103       s.colpo_fatale = 5;
00104       s.danno = 2;
00105       s.ricompensa_monete = 4;
00106     }
00107     else if (dado == 5)
00108     {
00109       sprintf(s.nome, "Vampiro Superiore");
00110       s.tipo = STANZA_NEMICO;
00111       s.colpo_fatale = 4;
00112       s.danno = 4;
00113       s.ricompensa_monete = 7;
00114       s.is_boss = 1;
00115     }
00116     else if (dado == 6)
00117     {
00118       sprintf(s.nome, "Demone Custode");
00119       s.tipo = STANZA_NEMICO;
00120       s.colpo_fatale = 4;
00121       s.danno = 6;
00122       s.ricompensa_monete = 10;
00123       s.is_boss = 1;
00124     }
00125   }
00126   // Missione 3: Grotta
00127   else if (tipo_missione == 3)
00128   {
00129     if (dado == 1)
00130     {
00131       sprintf(s.nome, "Stanza Vuota");
00132       s.tipo = STANZA_VUOTA;
00133     }
00134     else if (dado == 2)
00135     {
00136       sprintf(s.nome, "Cristalli Cadenti");
00137       s.tipo = STANZA_TRAPPOLA;
00138       s.danno = 2;
00139     }
00140     else if (dado == 3)
00141     {
```

```
00142       sprintf(s.nome, "Ponte Pericolante");
00143       s.tipo = STANZA_TRAPPOLA;
00144       s.danno = 0;
00145     }
00146     else if (dado == 4)
00147     {
00148       sprintf(s.nome, "Forziere Misterioso");
00149       s.tipo = STANZA_TRAPPOLA;
00150       s.danno = 2;
00151       s.ricompensa_monete = 10;
00152     }
00153     else if (dado == 5)
00154     {
00155       sprintf(s.nome, "Rupe scoscesa");
00156       s.tipo = STANZA_TRAPPOLA;
00157       s.danno = lancia_dado(6);
00158     }
00159     else if (dado == 6)
00160     {
00161       sprintf(s.nome, "Drago Antico");
00162       s.tipo = STANZA_NEMICO;
00163       s.colpo_fatale = 5;
00164       s.danno = 10;
00165       s.ricompensa_monete = 12;
00166       s.is_boss = 1;
00167     }
00168   }
00169   return s;
00170 }
00171
00172 int esegui_missione(Giocatore *g, int tipo_missione, const char *nome_missione)
00173 {
00174   int target = (tipo_missione == 1) ? 3 : 1;
00175   int progress = 0;
00176   int n_stanze = 0;
00177   int completata = 0;
00178
00179   do
00180   {
00181     pulisci_schermo();
00182     printf("=== %s ===\n", nome_missione);
00183
00184     // Obiettivi missione 'Palude'
00185     if (tipo_missione == 1)
00186       printf("Obiettivo: Sconfiggi %d Generali Orco (%d/%d)\n", target, progress, target);
00187
00188     // Obiettivi missione 'Magione'
00189     else if (tipo_missione == 2)
00190       printf("Obiettivo: Sconfiggi Vampiro Superiore (%d/%d) e prendi Chiave (%d/1)\n", progress,
      target, g->ha_chiave_castello);
00191
00192     // Obiettivi missione 'Grotta'
00193     else if (tipo_missione == 3)
00194       printf("Obiettivo: Sconfiggi Drago (%d/1) e prendi Spada (%d/1)\n", progress, g->ha_spada_eroe);
00195
00196     // Stanze esplorate
00197     printf("Stanze Esplorate: %d/10\n\n", n_stanze);
00198
00199     // Menu azioni
00200     if (n_stanze < 10)
00201       printf("1. Esplora\n");
00202     printf("2. Negozio\n3. Inventario\n4. Torna al Villaggio (Completa Obbiettivo oppure Paga 50
      monete)\n\n");
00203     printf("Seleziona una delle opzioni del menu [1-4]: ");
00204
00205     // Gestione scelta utente
00206     int scelta = leggi_intero();
00207
00208     // Scelta esplora
00209     if (scelta == 1 && n_stanze < 10)
00210     {
00211       n_stanze++;
00212       int dado = lancia_dado(6);
00213
00214       // Forza spawn boss se fine dungeon
00215       // Palude: Generale Orco
00216       if (n_stanze >= 8 && !completata && tipo_missione == 1)
00217         dado = 6;
00218
00219       // Magione: Demone Custode + Vampiro
00220       if (n_stanze >= 9 && !completata && tipo_missione == 2)
00221       {
00222         if (g->ha_chiave_castello == 0)
00223           dado = 5;
00224         else if (g->ha_chiave_castello == 1)
00225           dado = 6;
00226       }
```

```
00227       else if (completata && tipo_missione == 2)
00228         dado = lancia_dado(4); // evita di far spawnare di nuovo i boss
00229
00230       // Grotta: Drago Antico
00231       if (n_stanze >= 10 && !completata && tipo_missione == 3)
00232         dado = 6;
00233       else if (completata && tipo_missione == 3)
00234         dado = lancia_dado(5); // evita di far spawnare di nuovo il drago
00235
00236       Stanza s = genera_stanza_missione(tipo_missione, dado);
00237       printf("Incontri: %s\n", s.nome);
00238
00239       if (s.tipo == STANZA_NEMICO)
00240       {
00241         if (inizia_combattimento(g, &s))
00242         {
00243           // Update progressi missione 'Palude'
00244           if (tipo_missione == 1 && s.is_boss)
00245             progress++;
00246
00247           // Update progressi missione 'Magione'
00248           else if (tipo_missione == 2 && s.is_boss)
00249           {
00250             if (strcmp(s.nome, "Vampiro Superiore") == 0)
00251             {
00252               progress++;
00253             }
00254             else if (strcmp(s.nome, "Demone Custode") == 0)
00255             {
00256               printf("Trovata Chiave del Castello!\n");
00257               g->ha_chiave_castello = 1;
00258               g->numero_oggetti++;
00259             }
00260           }
00261           // Update progressi missione 'Grotta'
00262           else if (tipo_missione == 3 && s.is_boss)
00263           {
00264             printf("Trovata Spada Eroe!\n");
00265             g->ha_spada_eroe = 1;
00266             g->numero_oggetti++;
00267             progress++;
00268           }
00269         }
00270         else
00271           return 0; // Game Over
00272       }
00273       else if (s.tipo == STANZA_TRAPPOLA)
00274       {
00275         if (strcmp(s.nome, "Forziere Misterioso") == 0)
00276         {
00277           int chance = lancia_dado(2);
00278           printf("Apri il forziere... ");
00279           if (chance == 1)
00280           {
00281             printf("Trappola!\n");
00282             applica_danno_trappola(g, &s);
00283           }
00284           else
00285           {
00286             printf("Monete! (+%d)\n", s.ricompensa_monete);
00287             g->monete += s.ricompensa_monete;
00288           }
00289         }
00290         else if (strcmp(s.nome, "Ponte Pericolante") == 0)
00291         {
00292           g->monete -= 3;
00293           if (g->monete < 0)
00294             g->monete = 0;
00295           printf("Caduto dal ponte! Perdi 3 monete. (Monete: %d)\n", g->monete);
00296         }
00297         else if (s.danno > 0)
00298           applica_danno_trappola(g, &s);
00299       }
00300
00301       // Check completamento missione 'Palude'
00302       if (tipo_missione == 1 && progress >= target)
00303         completata = 1;
00304       // Check completamento missione 'Magione'
00305       if (tipo_missione == 2 && progress >= target && g->ha_chiave_castello == 1)
00306         completata = 1;
00307       // Check completamento missione 'Grotta'
00308       if (tipo_missione == 3 && progress >= target && g->ha_spada_eroe == 1)
00309         completata = 1;
00310
00311       if (g->punti_vita <= 0)
00312         return 0;
00313
```

```
00314       printf("Premi Invio...");
00315       getchar();
00316     }
00317
00318     // Scelta negozio
00319     else if (scelta == 2)
00320       mostra_negozio(g);
00321
00322     // Scelta inventario
00323     else if (scelta == 3)
00324     {
00325       stampa_statistiche_giocatore(g);
00326       getchar();
00327     }
00328
00329     // Scelta fuga
00330     else if (scelta == 4)
00331     {
00332       if (completata)
00333       {
00334         printf("\nMISSIONE COMPIUTA!\n");
00335         getchar();
00336         return 1;
00337       }
00338       if (g->monete >= 50)
00339       {
00340         g->monete -= 50;
00341         return 0;
00342       }
00343       printf("Non hai 50 monete!\n");
00344       getchar();
00345     }
00346   } while (1);
00347 }
```

6.7 Riferimenti per il file src/dungeon.h

Gestione della generazione e del flusso del dungeon.

```
#include "stanza.h"
#include "giocatore.h"
```

Grafo delle dipendenze di inclusione per dungeon.h: Questo grafo mostra quali altri file includono direttamente o indirettamente questo file:

Funzioni

- Stanza genera_stanza_missione (int tipo_missione, int dado)

  Genera una stanza casuale in base alla missione e al lancio del dado.
- int esegui_missione (Giocatore *g, int tipo_missione, const char *nome_missione)

  Gestisce il loop principale di esecuzione di una missione.

6.7.1 Descrizione dettagliata

Gestione della generazione e del flusso del dungeon.

Definizione nel file dungeon.h.

6.7.2 Documentazione delle funzioni

6.7.2.1 esegui_missione()

```
int esegui_missione (
            Giocatore * g,
            int tipo_missione,
            const char * nome_missione)
```

Parametri

g	Puntatore al giocatore.
tipo_missione	ID della missione da eseguire.
nome_missione	Nome visualizzato della missione.

Restituisce

1 se la missione è completata con successo, 0 in caso di fallimento o rinuncia.

Definizione alla linea 172 del file dungeon.c.

Referenzia applica_danno_trappola(), Stanza::danno, genera_stanza_missione(), Giocatore::ha_chiave_castello, Giocatore::ha_spada_eroe, inizia_combattimento(), Stanza::is_boss, lancia_dado(), leggi_intero(), Giocatore::monete, mostra_negozio(), Stanza::nome, Giocatore::numero_oggetti, pulisci_schermo(), Giocatore::punti_vita, Stanza::ricompensa_monete, stampa_statistiche_giocatore(), STANZA_NEMICO, STANZA_TRAPPOLA, e Stanza::tipo.

Referenziato da mostra_menu_missione().

6.7.2.2 genera_stanza_missione()

```
Stanza genera_stanza_missione (
            int tipo_missione,
            int dado)
```

Genera una stanza casuale in base alla missione e al lancio del dado.

Parametri

tipo_missione	ID della missione corrente (1, 2 o 3).
dado	Risultato del lancio del dado per determinare il contenuto della stanza.

Restituisce

La struct Stanza generata.

Definizione alla linea 12 del file dungeon.c.

Referenzia Stanza::colpo_fatale, Stanza::danno, Stanza::is_boss, lancia_dado(), Stanza::nome, Stanza::ricompensa_monete, STANZA_NEMICO, STANZA_NESSUNA, STANZA_TRAPPOLA, STANZA_VUOTA, e Stanza::tipo.

Referenziato da esegui_missione().

6.8 dungeon.h

Vai alla documentazione di questo file.

```
00001
00005
00006 #ifndef DUNGEON_H
00007 #define DUNGEON_H
00008
00009 #include "stanza.h"
00010 #include "giocatore.h"
00011
00019 Stanza genera_stanza_missione(int tipo_missione, int dado);
00020
00032 int esegui_missione(Giocatore *g, int tipo_missione, const char *nome_missione);
00033
00034 #endif
```

6.9 Riferimenti per il file src/giocatore.c

Implementazione delle funzioni relative al giocatore.

```
#include "giocatore.h"
#include <stdio.h>
```

Grafo delle dipendenze di inclusione per giocatore.c:

Funzioni

- void inizializza_giocatore (Giocatore *g)

  Inizializza un nuovo giocatore con valori di default.
- void stampa_statistiche_giocatore (const Giocatore *g)

  Stampa a video le statistiche e l'inventario del giocatore.

6.9.1 Descrizione dettagliata

Implementazione delle funzioni relative al giocatore.

Definizione nel file giocatore.c.

6.9.2 Documentazione delle funzioni

6.9.2.1 inizializza_giocatore()

```
void inizializza_giocatore (
            Giocatore * g)
```

Inizializza un nuovo giocatore con valori di default.

Imposta HP a 20, monete a 0 e resetta missioni e inventario.

Parametri

g	Puntatore alla struct Giocatore da inizializzare.

Definizione alla linea 9 del file giocatore.c.

Referenzia Giocatore::ha_armatura, Giocatore::ha_chiave_castello, Giocatore::ha_spada, Giocatore::ha_spada_eroe, Giocatore::max_punti_vita, Giocatore::missione_grotta, Giocatore::missione_magione, Giocatore::missione_palude, Giocatore::monete, Giocatore::numero_oggetti, e Giocatore::punti_vita.

Referenziato da mostra_menu_principale().

6.9.2.2 stampa_statistiche_giocatore()

```
void stampa_statistiche_giocatore (
            const Giocatore * g)
```


Stampa a video le statistiche e l'inventario del giocatore.

Mostra HP, monete, missioni completate ed equipaggiamento attuale.

Parametri

g	Puntatore alla struct Giocatore.

Definizione alla linea 24 del file giocatore.c.

Referenzia Giocatore::ha_armatura, Giocatore::ha_chiave_castello, Giocatore::ha_spada, Giocatore::ha_spada_eroe, Giocatore::max_punti_vita, Giocatore::missione_grotta, Giocatore::missione_magione, Giocatore::missione_palude, Giocatore::monete, Giocatore::numero_oggetti, e Giocatore::punti_vita.

Referenziato da esegui_missione(), e mostra_menu_villaggio().

6.10 giocatore.c

Vai alla documentazione di questo file.

```
00001
00005
00006 #include "giocatore.h"
00007 #include <stdio.h>
00008
00009 void inizializza_giocatore(Giocatore *g)
00010 {
00011   g->punti_vita = 20;
00012   g->max_punti_vita = 20;
00013   g->monete = 0;
00014   g->numero_oggetti = 0;
00015   g->missione_palude = 0;
00016   g->missione_magione = 0;
00017   g->missione_grotta = 0;
00018   g->ha_spada = 0;
00019   g->ha_armatura = 0;
00020   g->ha_spada_eroe = 0;
00021   g->ha_chiave_castello = 0;
00022 }
00023
00024 void stampa_statistiche_giocatore(const Giocatore *g)
00025 {
00026   printf("\n=== Inventario Giocatore ===\n");
00027
00028   printf("Punti Vita: %d / %d\n", g->punti_vita, g->max_punti_vita);
00029   printf("Monete: %d\n", g->monete);
00030   printf("Oggetti: %d\n", g->numero_oggetti);
00031   printf("Missioni Completate: %d\n", g->missione_palude + g->missione_magione + g->missione_grotta);
00032   printf("Equipaggiamento:\n");
00033   if (g->ha_spada || g->ha_spada_eroe)
00034     printf("- Spada: %s (+%d attacco)\n",
00035            g->ha_spada_eroe ? "Spada dell'Eroe" : "Spada Ferro",
00036            g->ha_spada_eroe ? 2 : 1);
00037   else
00038     printf("- Nessuna arma\n");
00039   if (g->ha_armatura)
00040     printf("- Armatura: Si (-1 danno subito)\n");
00041   else
00042     printf("- Nessuna armatura\n");
00043   if (g->ha_chiave_castello)
00044     printf("- Oggetti Chiave: Chiave del Castello\n");
00045
00046   printf("============================\n");
00047 }
```

6.11 Riferimenti per il file src/giocatore.h

Definizione della struttura Giocatore e funzioni correlate.

Questo grafo mostra quali altri file includono direttamente o indirettamente questo file:

Strutture dati

- struct Giocatore

  Struttura che rappresenta lo stato del giocatore.

Funzioni

- void inizializza_giocatore (Giocatore *g)

  Inizializza un nuovo giocatore con valori di default.
- void stampa_statistiche_giocatore (const Giocatore *g)

  Stampa a video le statistiche e l'inventario del giocatore.

6.11.1 Descrizione dettagliata

Definizione della struttura Giocatore e funzioni correlate.

Definizione nel file giocatore.h.

6.11.2 Documentazione delle funzioni

6.11.2.1 inizializza_giocatore()

```
void inizializza_giocatore (
            Giocatore * g)
```

Inizializza un nuovo giocatore con valori di default.

Imposta HP a 20, monete a 0 e resetta missioni e inventario.

Parametri

g	Puntatore alla struct Giocatore da inizializzare.

Definizione alla linea 9 del file giocatore.c.

Referenzia Giocatore::ha_armatura, Giocatore::ha_chiave_castello, Giocatore::ha_spada, Giocatore::ha_spada_eroe, Giocatore::max_punti_vita, Giocatore::missione_grotta, Giocatore::missione_magione, Giocatore::missione_palude, Giocatore::monete, Giocatore::numero_oggetti, e Giocatore::punti_vita.

Referenziato da mostra_menu_principale().

6.11.2.2 stampa_statistiche_giocatore()

```
void stampa_statistiche_giocatore (
            const Giocatore * g)
```

Stampa a video le statistiche e l'inventario del giocatore.

Mostra HP, monete, missioni completate ed equipaggiamento attuale.

Parametri

g	Puntatore alla struct Giocatore.

Definizione alla linea 24 del file giocatore.c.

Referenzia Giocatore::ha_armatura, Giocatore::ha_chiave_castello, Giocatore::ha_spada, Giocatore::ha_spada_eroe, Giocatore::max_punti_vita, Giocatore::missione_grotta, Giocatore::missione_magione, Giocatore::missione_palude, Giocatore::monete, Giocatore::numero_oggetti, e Giocatore::punti_vita.

Referenziato da esegui_missione(), e mostra_menu_villaggio().

6.12 giocatore.h

Vai alla documentazione di questo file.

```
00001
00005
00006 #ifndef GIOCATORE_H
00007 #define GIOCATORE_H
00008
00012 typedef struct
00013 {
00014   int punti_vita;
00015   int max_punti_vita;
00016   int monete;
00017   int numero_oggetti;
00018
00019   // Flag per missioni completate
00020   int missione_palude;
00021   int missione_magione;
00022   int missione_grotta;
00023
00024   // Flag per oggetti chiave
00025   int ha_spada;
00026   int ha_armatura;
00027   int ha_spada_eroe;
00028   int ha_chiave_castello;
00029 } Giocatore;
00030
00038 void inizializza_giocatore(Giocatore *g);
00039
00047 void stampa_statistiche_giocatore(const Giocatore *g);
00048
00049 #endif
```

6.13 Riferimenti per il file src/main.c

Entry point del gioco CDungeon.

```
#include "giocatore.h"
#include "menu.h"
#include "salvataggio.h"
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
```

Grafo delle dipendenze di inclusione per main.c:

Funzioni

- int main ()

  Funzione principale del programma.

6.13.1 Descrizione dettagliata

Entry point del gioco CDungeon.

Definizione nel file main.c.

6.13.2 Documentazione delle funzioni

6.13.2.1 main()

```
int main ()
```

Funzione principale del programma.

Inizializza il generatore di numeri casuali, carica i salvataggi e avvia il loop principale del menu.

Restituisce

0 se il programma termina correttamente.

Definizione alla linea 20 del file main.c.

Referenzia carica_salvataggi_da_file(), libera_salvataggi(), e mostra_menu_principale().

6.14 main.c

Vai alla documentazione di questo file.

```
00001
00005 #include "giocatore.h"
00006 #include "menu.h"
00007 #include "salvataggio.h"
00008 #include <stdio.h>
00009 #include <stdlib.h>
00010 #include <time.h>
00011
00020 int main(){
00021   NodoSalvataggio *lista_salvataggi = NULL;
00022   Giocatore giocatore_corrente;
00023
00024   // Carica i salvataggi esistenti da file
00025   carica_salvataggi_da_file(&lista_salvataggi);
00026
00027   // Inizializza seed random
00028   srand(time(NULL));
00029
00030   printf("Benvenuto in CDungeon! (premi Invio per iniziare)\n");
00031   getchar();
00032   mostra_menu_principale(&lista_salvataggi, &giocatore_corrente);
00033
00034   // Pulizia della memoria alla chiusura
00035   libera_salvataggi(lista_salvataggi);
00036
00037   return 0;
00038 }
```

6.15 Riferimenti per il file src/menu.c

Implementazione dei menu e dell'interfaccia utente.

```
#include "menu.h"
#include "combattimento.h"
#include "dungeon.h"
#include "giocatore.h"
#include "salvataggio.h"
#include "utilita.h"
#include <ctype.h>
#include <stdio.h>
#include <string.h>
```

Grafo delle dipendenze di inclusione per menu.c:

Funzioni

- void gestisci_trucchi (Giocatore *g, NodoSalvataggio **lista)

  Gestisce i trucchi del gioco.
- void mostra_menu_salvataggi (Giocatore *g, NodoSalvataggio **lista)

  Mostra il menu dei salvataggi.
- void mostra_menu_principale (NodoSalvataggio **lista, Giocatore *g)

  Mostra il menu principale del gioco.
- void mostra_menu_villaggio (Giocatore *g, NodoSalvataggio **lista)

  Mostra il menu del villaggio.
- void mostra_menu_missione (Giocatore *g)

  Mostra il menu di selezione delle missioni.
- void mostra_negozio (Giocatore *g)

  Mostra il negozio del villaggio.

6.15.1 Descrizione dettagliata

Implementazione dei menu e dell'interfaccia utente.

Definizione nel file menu.c.

6.15.2 Documentazione delle funzioni

6.15.2.1 gestisci_trucchi()

```
void gestisci_trucchi (
            Giocatore * g,
            NodoSalvataggio ** lista_salvataggi)
```

Gestisce i trucchi del gioco.

Permette di modificare i salvataggi esistenti per ottenere vantaggi come monete aggiuntive, HP aggiuntivi, o altri benefici.

Parametri

g	Puntatore al giocatore, necessario per caricare e modificare il salvataggio selezionato.
lista_salvataggi	Puntatore alla lista dei salvataggi per la selezione e modifica.

Definizione alla linea 316 del file menu.c.

Referenzia carica_salvataggio(), NodoSalvataggio::dati_giocatore, leggi_intero(), Giocatore::max_punti_vita, Giocatore::missione_grotta, Giocatore::missione_magione, Giocatore::missione_palude, Giocatore::monete, pulisci_schermo(), Giocatore::punti_vita, salva_tutto_su_file(), e stampa_salvataggi().

Referenziato da mostra_menu_principale().

6.15.2.2 mostra_menu_missione()

```
void mostra_menu_missione (
            Giocatore * g)
```

Mostra il menu di selezione delle missioni.

Elenca le missioni disponibili e avviabili dal giocatore.

Parametri

g	Puntatore al giocatore.

Definizione alla linea 179 del file menu.c.

Referenzia combattimento_boss_finale(), esegui_missione(), leggi_intero(), Giocatore::missione_grotta, Giocatore::missione_magion
Giocatore::missione_palude, pulisci_schermo(), e Giocatore::punti_vita.

Referenziato da mostra_menu_villaggio().

6.15.2.3 mostra_menu_principale()

```
void mostra_menu_principale (
            NodoSalvataggio ** lista_salvataggi,
            Giocatore * giocatore_corrente)
```

Mostra il menu principale del gioco.

Permette di iniziare una nuova partita, caricare un salvataggio o uscire. Contiene anche la gestione dei trucchi tramite codice segreto.

Parametri

lista_salvataggi	Puntatore alla lista dei salvataggi.
giocatore_corrente	Puntatore alla struct Giocatore corrente.

Definizione alla linea 20 del file menu.c.

Referenzia gestisci_trucchi(), inizializza_giocatore(), leggi_input_char(), mostra_menu_salvataggi(), mostra_menu_villaggio(), e pulisci_schermo().

Referenziato da main(), e mostra_menu_villaggio().

6.15.2.4 mostra_menu_salvataggi()

```
void mostra_menu_salvataggi (
            Giocatore * g,
            NodoSalvataggio ** lista_salvataggi)
```

Mostra il menu dei salvataggi.

Elenca i salvataggi disponibili e permette di caricarli o eliminarli.

Parametri

g	Puntatore al giocatore, necessario per caricare il salvataggio selezionato.
lista_salvataggi	Puntatore alla lista dei salvataggi.

Definizione alla linea 84 del file menu.c.

Referenzia carica_salvataggio(), NodoSalvataggio::dati_giocatore, elimina_salvataggio(), leggi_input_char(), leggi_intero(), mostra_menu_villaggio(), e stampa_salvataggi().

Referenziato da mostra_menu_principale().

6.15.2.5 mostra_menu_villaggio()

```
void mostra_menu_villaggio (
            Giocatore * g,
            NodoSalvataggio ** lista_salvataggi)
```

Mostra il menu del villaggio.

Permette di curarsi, gestire l'inventario, salvare o intraprendere nuove missioni.

Parametri

g	Puntatore al giocatore.
lista_salvataggi	Puntatore alla lista dei salvataggi per il salvataggio manuale.

Definizione alla linea 132 del file menu.c.

Referenzia aggiungi_salvataggio(), leggi_input_char(), leggi_intero(), Giocatore::max_punti_vita, Giocatore::monete, mostra_menu_missione(), mostra_menu_principale(), pulisci_schermo(), Giocatore::punti_vita, e stampa_statistiche_giocatore().

Referenziato da mostra_menu_principale(), e mostra_menu_salvataggi().

6.15.2.6 mostra_negozio()

```
void mostra_negozio (
            Giocatore * g)
```

Mostra il negozio del villaggio.

Permette al giocatore di acquistare oggetti e cure in cambio di monete.

Parametri

g	Puntatore al giocatore.

Definizione alla linea 252 del file menu.c.

Referenzia Giocatore::ha_armatura, Giocatore::ha_spada, lancia_dado(), leggi_intero(), Giocatore::max_punti_vita, Giocatore::monete, Giocatore::numero_oggetti, pulisci_schermo(), e Giocatore::punti_vita.

Referenziato da esegui_missione().

6.16 menu.c

Vai alla documentazione di questo file.

```
00001
00005
00006 #include "menu.h"
00007 #include "combattimento.h"
00008 #include "dungeon.h"
00009 #include "giocatore.h"
00010 #include "salvataggio.h"
00011 #include "utilita.h"
00012 #include <ctype.h>
00013 #include <stdio.h>
00014 #include <string.h>
00015
00016 // Forward declarations
00017 void gestisci_trucchi(Giocatore *g, NodoSalvataggio **lista);
00018 void mostra_menu_salvataggi(Giocatore *g, NodoSalvataggio **lista);
00019
00020 void mostra_menu_principale(NodoSalvataggio **lista, Giocatore *g)
00021 {
00022   char in;
00023   const char *k_code = "wwssadadba ";
00024   int k_idx = 0;
00025   int cheat = 0;
00026
00027   do
00028   {
00029     pulisci_schermo();
00030     printf("=== CDungeon ===\n\n");
00031
00032     printf("1. Nuova Partita\n2. Carica Salvataggio\n");
00033     if (cheat)
00034       printf("3. Trucchi\n4. Esci\n");
00035     else
00036       printf("3. Esci\n\n");
00037
00038     printf("Seleziona una delle opzioni del menu [1-%d]: ", cheat ? 4 : 3);
00039
00040     in = leggi_input_char();
00041     if (isdigit(in))
00042     {
00043       // Converto il carattere in un intero
00044       int scelta = in - '0';
00045
00046       // Gestione delle opzioni del menu
00047       if (scelta == 1)
00048       {
00049         inizializza_giocatore(g);
00050         mostra_menu_villaggio(g, lista);
00051       }
00052       else if (scelta == 2)
00053       {
00054         mostra_menu_salvataggi(g, lista);
00055       }
00056       else if (cheat && scelta == 3)
00057         gestisci_trucchi(g, lista);
00058       else
00059         return;
00060     }
00061
00062     // Gestione del codice segreto per i trucchi
00063     else
00064     {
```

```
00065       if (in == k_code[k_idx])
00066       {
00067         k_idx++;
00068         if (k_idx == strlen(k_code))
00069         {
00070           cheat = 1;
00071           k_idx = 0;
00072           printf("Trucchi Attivi!\n");
00073           getchar();
00074         }
00075       }
00076       else
00077       {
00078         k_idx = (in == 'w') ? 1 : 0;
00079       }
00080     }
00081   } while (1);
00082 }
00083
00084 void mostra_menu_salvataggi(Giocatore *g, NodoSalvataggio **lista)
00085 {
00086   stampa_salvataggi(*lista);
00087
00088   if (!*lista)
00089   {
00090     printf("Premi Invio per tornare al menu principale...");
00091     getchar();
00092     return;
00093   }
00094
00095   printf("ID: ");
00096   int id = leggi_intero();
00097   if (id)
00098   {
00099     NodoSalvataggio *salv = carica_salvataggio(*lista, id);
00100     if (salv)
00101     {
00102       printf("\nScegli operazione:\n");
00103       printf("1. Carica\n2. Elimina\n0. Annulla\nScelta: ");
00104       int op = leggi_intero();
00105       if (op == 1)
00106       {
00107         *g = salv->dati_giocatore;
00108         printf("Salvataggio caricato.\n");
00109         getchar();
00110         mostra_menu_villaggio(g, lista);
00111       }
00112       else if (op == 2)
00113       {
00114         printf("Sei sicuro di voler eliminare il salvataggio? (s/n): ");
00115         char conferma = leggi_input_char();
00116         if (conferma == 's')
00117         {
00118           elimina_salvataggio(lista, id);
00119         }
00120         getchar(); // Attendi invio per leggere messaggio
00121       }
00122     }
00123     else
00124     {
00125       printf("Salvataggio non trovato.\n");
00126       getchar();
00127       // mostra_menu_salvataggi(g, lista);
00128     }
00129   }
00130 }
00131
00132 void mostra_menu_villaggio(Giocatore *g, NodoSalvataggio **lista)
00133 {
00134   do
00135   {
00136     if (g->punti_vita <= 0)
00137     {
00138       mostra_menu_principale(lista, g);
00139       return;
00140     }
00141
00142     pulisci_schermo();
00143     printf("=== Menù Villaggio ===\nHP: %d/%d | Monete: %d\n", g->punti_vita, g->max_punti_vita,
      g->monete);
00144
00145     printf("\n1. Intraprendi una missione\n2. Riposati\n3. Inventario\n4. Salva la partita\n5.
      Esci\n\n");
00146
00147     printf("Seleziona una delle opzioni del menù [1-5]: ");
00148
00149     // Gestione delle opzioni del menu
```

```
00150     int s = leggi_intero();
00151     switch (s)
00152     {
00153     case 1:
00154       mostra_menu_missione(g);
00155       break;
00156     case 2:
00157       g->punti_vita = g->max_punti_vita;
00158       printf("Riposato.\n");
00159       getchar();
00160       break;
00161     case 3:
00162       stampa_statistiche_giocatore(g);
00163       getchar();
00164       break;
00165     case 4:
00166       aggiungi_salvataggio(lista, g);
00167       getchar();
00168       break;
00169     case 5:
00170       printf("Stai uscendo dal gioco, ricodati di salvare la partita per non perdere i progressi!. Sei
      sicuro di voler uscire? (s/n): ");
00171       char conferma = leggi_input_char();
00172       if (conferma == 's')
00173         return;
00174       break;
00175     }
00176   } while (1);
00177 }
00178
00179 void mostra_menu_missione(Giocatore *g)
00180 {
00181   do
00182   {
00183     pulisci_schermo();
00184     printf("=== Menù di Selezione Missioni ===\n\n");
00185
00186     // Flag di controllo di quali missioni sono completate
00187     int f1 = g->missione_palude;
00188     int f2 = g->missione_magione;
00189     int f3 = g->missione_grotta;
00190
00191     if (!f1)
00192       printf("1. Palude Putrescente\n");
00193     if (!f2)
00194       printf("2. Magione Infestata\n");
00195     if (!f3)
00196       printf("3. Grotta di Cristallo\n");
00197     if (f1 && f2 && f3)
00198       printf("4. Castello del Signore Oscuro (Boss Finale)\n");
00199
00200     printf("0. Indietro\n\n");
00201
00202     if (!f1 || !f2 || !f3)
00203       printf("Seleziona una delle opzioni del menu [1-3]: ");
00204     else if (f1 && f2 && f3)
00205       printf("Seleziona una delle opzioni del menu [4]: ");
00206
00207     int s = leggi_intero();
00208     if (s == 0)
00209       return;
00210
00211     // Esegui la missione selezionata
00212     if (s == 1 && !f1)
00213     {
00214       if (esegui_missione(g, 1, "Palude"))
00215         g->missione_palude = 1;
00216       else if (g->punti_vita <= 0)
00217         return;
00218     }
00219     else if (s == 2 && !f2)
00220     {
00221       if (esegui_missione(g, 2, "Magione"))
00222         g->missione_magione = 1;
00223       else if (g->punti_vita <= 0)
00224         return;
00225     }
00226     else if (s == 3 && !f3)
00227     {
00228       if (esegui_missione(g, 3, "Grotta"))
00229         g->missione_grotta = 1;
00230       else if (g->punti_vita <= 0)
00231         return;
00232     }
00233     else if (s == 4 && f1 && f2 && f3)
00234     {
00235       if (combattimento_boss_finale(g))
```

```
00236       {
00237         printf("\nHAI VINTO IL GIOCO!\n");
00238         getchar();
00239         return;
00240       }
00241       else
00242       {
00243         printf("\nSEI STATO SCONFITTO!\n");
00244         g->punti_vita = 0;
00245         getchar();
00246         return;
00247       }
00248     }
00249   } while (1);
00250 }
00251
00252 void mostra_negozio(Giocatore *g)
00253 {
00254   do
00255   {
00256     pulisci_schermo();
00257     printf("=== Negozio (Monete: %d) ===\n", g->monete);
00258     printf("Oggetto \t| Descrizione \t\t\t| Costo\n");
00259     printf("----------------|-------------------------------|-------\n");
00260     printf("1. Cura \t| Ripristina punti vita (1-6) \t| 4 monete\n");
00261     if (!g->ha_spada)
00262       printf("2. Spada \t| Aumenta attacco (+1) \t\t| 5 monete\n");
00263     if (!g->ha_armatura)
00264       printf("3. Armatura \t| Riduce danno subito (-1) \t| 10 monete\n");
00265     printf("----------------|-------------------------------|-------\n");
00266     printf("0. Esci\n\n");
00267     printf("Seleziona un oggetto da acquistare: ");
00268
00269     // Gestione delle opzioni del menu
00270     int scelta = leggi_intero();
00271     if (scelta == 0)
00272       return;
00273
00274     if (scelta == 1 && g->monete >= 4)
00275     {
00276       g->monete -= 4;
00277
00278       int punti_ripristinati = lancia_dado(6);
00279       g->punti_vita += punti_ripristinati;
00280       if (g->punti_vita > g->max_punti_vita)
00281         g->punti_vita = g->max_punti_vita;
00282       printf("Ripristinati %d punti vita.\n", punti_ripristinati);
00283     }
00284     else if (scelta == 2 && g->monete >= 5)
00285     {
00286       if (!g->ha_spada)
00287       {
00288         g->monete -= 5;
00289         g->ha_spada = 1;
00290         g->numero_oggetti++;
00291         printf("Spada acquistata.\n");
00292       }
00293     }
00294     else if (scelta == 3 && g->monete >= 10)
00295     {
00296       if (!g->ha_armatura)
00297       {
00298         g->monete -= 10;
00299         g->ha_armatura = 1;
00300         g->numero_oggetti++;
00301         printf("Armatura acquistata.\n");
00302       }
00303     }
00304     else if (scelta < 1 || scelta > 3)
00305     {
00306       printf("Scelta non valida.\n");
00307     }
00308     else
00309     {
00310       printf("Non hai abbastanza monete.\n");
00311     }
00312     getchar();
00313   } while (1);
00314 }
00315
00316 void gestisci_trucchi(Giocatore *g, NodoSalvataggio **lista)
00317 {
00318   do
00319   {
00320     pulisci_schermo();
00321     stampa_salvataggi(*lista);
00322     printf("Inserisci ID salvataggio (0 per uscire): ");
```

```
00323     int id = leggi_intero();
00324     if (id == 0)
00325       return;
00326
00327     NodoSalvataggio *salv = carica_salvataggio(*lista, id);
00328     if (!salv)
00329     {
00330       printf("Salvataggio non trovato.\n");
00331       getchar();
00332       continue;
00333     }
00334
00335     // Copia i dati dal salvataggio per lavorarci
00336     *g = salv->dati_giocatore;
00337
00338     printf("\n--- TRUCCHI (Salvataggio ID: %d) ---\n", id);
00339     printf("1. Max Monete (999)\n");
00340     printf("2. Max HP (999)\n");
00341     printf("3. Sblocca Tutte le Missioni\n");
00342     printf("4. Torna indietro\n");
00343     printf("Scelta: ");
00344
00345     int scelta = leggi_intero();
00346     int modificato = 0;
00347
00348     // Applica i trucchi selezionati
00349     if (scelta == 1)
00350     {
00351       salv->dati_giocatore.monete = 999;
00352       modificato = 1;
00353       printf("Monete impostate a 999!\n");
00354     }
00355     else if (scelta == 2)
00356     {
00357       salv->dati_giocatore.max_punti_vita = 999;
00358       salv->dati_giocatore.punti_vita = 999;
00359       modificato = 1;
00360       printf("HP impostati a 999!\n");
00361     }
00362     else if (scelta == 3)
00363     {
00364       salv->dati_giocatore.missione_palude = 1;
00365       salv->dati_giocatore.missione_magione = 1;
00366       salv->dati_giocatore.missione_grotta = 1;
00367       modificato = 1;
00368       printf("Tutte le missioni sbloccate!\n");
00369     }
00370     else if (scelta == 4)
00371     {
00372       continue;
00373     }
00374
00375     // Se sono state apportate modifiche, salva il salvataggio aggiornato su file
00376     if (modificato)
00377     {
00378       salva_tutto_su_file(*lista);
00379       printf("Salvataggio aggiornato su file.\n");
00380       // Aggiorna anche g per riflettere le modifiche se l'utente carica subito
00381       *g = salv->dati_giocatore;
00382     }
00383     getchar();
00384
00385   } while (1);
00386 }
```

6.17 Riferimenti per il file src/menu.h

Gestione dei menu e dell'interfaccia utente.

```
#include "giocatore.h"
#include "salvataggio.h"
```

Grafo delle dipendenze di inclusione per menu.h: Questo grafo mostra quali altri file includono direttamente o indirettamente questo file:

Funzioni

- void mostra_menu_principale (NodoSalvataggio **lista_salvataggi, Giocatore *giocatore_corrente)

Mostra il menu principale del gioco.

- void mostra_menu_salvataggi (Giocatore *g, NodoSalvataggio **lista_salvataggi)

  Mostra il menu dei salvataggi.
- void mostra_menu_villaggio (Giocatore *g, NodoSalvataggio **lista_salvataggi)

  Mostra il menu del villaggio.
- void mostra_menu_missione (Giocatore *g)

  Mostra il menu di selezione delle missioni.
- void mostra_negozio (Giocatore *g)

  Mostra il negozio del villaggio.
- void gestisci_trucchi (Giocatore *g, NodoSalvataggio **lista_salvataggi)

  Gestisce i trucchi del gioco.

6.17.1 Descrizione dettagliata

Gestione dei menu e dell'interfaccia utente.

Definizione nel file menu.h.

6.17.2 Documentazione delle funzioni

6.17.2.1 gestisci_trucchi()

```
void gestisci_trucchi (
            Giocatore * g,
            NodoSalvataggio ** lista_salvataggi)
```

Gestisce i trucchi del gioco.

Permette di modificare i salvataggi esistenti per ottenere vantaggi come monete aggiuntive, HP aggiuntivi, o altri benefici.

Parametri

g	Puntatore al giocatore, necessario per caricare e modificare il salvataggio selezionato.
lista_salvataggi	Puntatore alla lista dei salvataggi per la selezione e modifica.

Definizione alla linea 316 del file menu.c.

Referenzia carica_salvataggio(), NodoSalvataggio::dati_giocatore, leggi_intero(), Giocatore::max_punti_vita, Giocatore::missione_grotta, Giocatore::missione_magione, Giocatore::missione_palude, Giocatore::monete, pulisci_schermo(), Giocatore::punti_vita, salva_tutto_su_file(), e stampa_salvataggi().

Referenziato da mostra_menu_principale().

6.17.2.2 mostra_menu_missione()

```
void mostra_menu_missione (
            Giocatore * g)
```


Elenca le missioni disponibili e avviabili dal giocatore.

Parametri

g	Puntatore al giocatore.

Definizione alla linea 179 del file menu.c.

Referenzia combattimento_boss_finale(), esegui_missione(), leggi_intero(), Giocatore::missione_grotta, Giocatore::missione_magion
Giocatore::missione_palude, pulisci_schermo(), e Giocatore::punti_vita.

Referenziato da mostra_menu_villaggio().

6.17.2.3 mostra_menu_principale()

```
void mostra_menu_principale (
            NodoSalvataggio ** lista_salvataggi,
            Giocatore * giocatore_corrente)
```

Mostra il menu principale del gioco.

Permette di iniziare una nuova partita, caricare un salvataggio o uscire. Contiene anche la gestione dei trucchi tramite codice segreto.

Parametri

lista_salvataggi	Puntatore alla lista dei salvataggi.
giocatore_corrente	Puntatore alla struct Giocatore corrente.

Definizione alla linea 20 del file menu.c.

Referenzia gestisci_trucchi(), inizializza_giocatore(), leggi_input_char(), mostra_menu_salvataggi(), mostra_menu_villaggio(), e pulisci_schermo().

Referenziato da main(), e mostra_menu_villaggio().

6.17.2.4 mostra_menu_salvataggi()

```
void mostra_menu_salvataggi (
            Giocatore * g,
            NodoSalvataggio ** lista_salvataggi)
```

Mostra il menu dei salvataggi.

Elenca i salvataggi disponibili e permette di caricarli o eliminarli.

Parametri

g	Puntatore al giocatore, necessario per caricare il salvataggio selezionato.
lista_salvataggi	Puntatore alla lista dei salvataggi.

Definizione alla linea 84 del file menu.c.

Referenzia carica_salvataggio(), NodoSalvataggio::dati_giocatore, elimina_salvataggio(), leggi_input_char(), leggi_intero(), mostra_menu_villaggio(), e stampa_salvataggi().

Referenziato da mostra_menu_principale().

6.17.2.5 mostra_menu_villaggio()

```
void mostra_menu_villaggio (
            Giocatore * g,
            NodoSalvataggio ** lista_salvataggi)
```

Mostra il menu del villaggio.

Permette di curarsi, gestire l'inventario, salvare o intraprendere nuove missioni.

Parametri

g	Puntatore al giocatore.
lista_salvataggi	Puntatore alla lista dei salvataggi per il salvataggio manuale.

Definizione alla linea 132 del file menu.c.

Referenzia aggiungi_salvataggio(), leggi_input_char(), leggi_intero(), Giocatore::max_punti_vita, Giocatore::monete, mostra_menu_missione(), mostra_menu_principale(), pulisci_schermo(), Giocatore::punti_vita, e stampa_statistiche_giocatore().

Referenziato da mostra_menu_principale(), e mostra_menu_salvataggi().

6.17.2.6 mostra_negozio()

```
void mostra_negozio (
            Giocatore * g)
```

Mostra il negozio del villaggio.

Permette al giocatore di acquistare oggetti e cure in cambio di monete.

Parametri

g	Puntatore al giocatore.

Definizione alla linea 252 del file menu.c.

Referenzia Giocatore::ha_armatura, Giocatore::ha_spada, lancia_dado(), leggi_intero(), Giocatore::max_punti_vita, Giocatore::monete, Giocatore::numero_oggetti, pulisci_schermo(), e Giocatore::punti_vita.

Referenziato da esegui_missione().

6.18 menu.h

Vai alla documentazione di questo file.

```
00001
00005
00006 #ifndef MENU_H
00007 #define MENU_H
00008
00009 #include "giocatore.h"
00010 #include "salvataggio.h"
00011
00021 void mostra_menu_principale(NodoSalvataggio **lista_salvataggi, Giocatore *giocatore_corrente);
00022
00031 void mostra_menu_salvataggi(Giocatore *g, NodoSalvataggio **lista_salvataggi);
00032
00041 void mostra_menu_villaggio(Giocatore *g, NodoSalvataggio **lista_salvataggi);
00042
00050 void mostra_menu_missione(Giocatore *g);
00051
00059 void mostra_negozio(Giocatore *g);
00060
00069 void gestisci_trucchi(Giocatore *g, NodoSalvataggio **lista_salvataggi);
00070
00071 #endif
```

6.19 Riferimenti per il file src/salvataggio.c

Implementazione delle funzioni per la gestione dei salvataggi.

```
#include "salvataggio.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
```

Grafo delle dipendenze di inclusione per salvataggio.c:

Funzioni

- void ottieni_data_corrente (char *buffer, size_t size)
- void salva_tutto_su_file (NodoSalvataggio *testa)

  Salva l'intera lista dei salvataggi su file.
- void aggiungi_salvataggio (NodoSalvataggio **testa, Giocatore *g)

  Aggiunge un nuovo salvataggio alla lista e al file.
- NodoSalvataggio * carica_salvataggio (NodoSalvataggio *testa, int id)

  Cerca un salvataggio nella lista tramite ID.
- void elimina_salvataggio (NodoSalvataggio **testa, int id)

  Elimina un salvataggio dalla lista e dal file.
- void stampa_salvataggi (NodoSalvataggio *testa)

  Stampa a video l'elenco dei salvataggi disponibili.
- void libera_salvataggi (NodoSalvataggio *testa)

  Libera la memoria allocata per la lista dei salvataggi.
- void carica_salvataggi_da_file (NodoSalvataggio **testa)

  Carica tutti i salvataggi dal file binario all'avvio.

6.19.1 Descrizione dettagliata

Implementazione delle funzioni per la gestione dei salvataggi.

Definizione nel file salvataggio.c.

6.19.2 Documentazione delle funzioni

6.19.2.1 aggiungi_salvataggio()

```
void aggiungi_salvataggio (
            NodoSalvataggio ** testa,
            Giocatore * g)
```

Aggiunge un nuovo salvataggio alla lista e al file.

Crea un nuovo nodo con i dati attuali del giocatore, genera un nuovo ID, lo aggiunge in testa alla lista e aggiorna il file di salvataggio.

Parametri

testa	Puntatore doppio alla testa della lista.
g	Puntatore ai dati del giocatore da salvare.

Definizione alla linea 44 del file salvataggio.c.

Referenzia NodoSalvataggio::dati_giocatore, NodoSalvataggio::id, ottieni_data_corrente(), NodoSalvataggio::prossimo, salva_tutto_su_file(), e NodoSalvataggio::timestamp.

Referenziato da mostra_menu_villaggio().

6.19.2.2 carica_salvataggi_da_file()

```
void carica_salvataggi_da_file (
            NodoSalvataggio ** testa)
```

Carica tutti i salvataggi dal file binario all'avvio.

Legge il file "salvataggi.bin" e ricostruisce la lista concatenata in memoria.

Parametri

testa	Puntatore doppio alla testa della lista che verrà inizializzata.

Definizione alla linea 165 del file salvataggio.c.

Referenzia NodoSalvataggio::dati_giocatore, NodoSalvataggio::id, libera_salvataggi(), NodoSalvataggio::prossimo, e NodoSalvataggio::timestamp.

Referenziato da main().

6.19.2.3 carica_salvataggio()

```
NodoSalvataggio * carica_salvataggio (
            NodoSalvataggio * testa,
            int id)
```

Cerca un salvataggio nella lista tramite ID.

Parametri

testa	Testa della lista dei salvataggi.
id	ID del salvataggio da cercare.

Restituisce

Puntatore al nodo trovato, o NULL se non esiste.

Definizione alla linea 79 del file salvataggio.c.

Referenzia NodoSalvataggio::id, e NodoSalvataggio::prossimo.

Referenziato da gestisci_trucchi(), e mostra_menu_salvataggi().

6.19.2.4 elimina_salvataggio()

```
void elimina_salvataggio (
            NodoSalvataggio ** testa,
            int id)
```

Elimina un salvataggio dalla lista e dal file.

Rimuove il nodo corrispondente all'ID specificato e aggiorna il file di salvataggio.

Parametri

testa	Puntatore doppio alla testa della lista.
id	ID del salvataggio da eliminare.

Definizione alla linea 95 del file salvataggio.c.

Referenzia NodoSalvataggio::id, NodoSalvataggio::prossimo, e salva_tutto_su_file().

Referenziato da mostra_menu_salvataggi().

6.19.2.5 libera_salvataggi()

```
void libera_salvataggi (
            NodoSalvataggio * testa)
```

Libera la memoria allocata per la lista dei salvataggi.

Da chiamare alla chiusura del programma.

Parametri

testa	Testa della lista da deallocare.

Definizione alla linea 151 del file salvataggio.c.

Referenzia NodoSalvataggio::prossimo.

Referenziato da carica_salvataggi_da_file(), e main().

6.19.2.6 ottieni_data_corrente()

```
void ottieni_data_corrente (
            char * buffer,
            size_t size)
```

Definizione alla linea 13 del file salvataggio.c.

Referenziato da aggiungi_salvataggio().

6.19.2.7 salva_tutto_su_file()

```
void salva_tutto_su_file (
```

Parametri

testa	Testa della lista dei salvataggi.

Definizione alla linea 21 del file salvataggio.c.

Referenzia NodoSalvataggio::dati_giocatore, NodoSalvataggio::id, NodoSalvataggio::prossimo, e NodoSalvataggio::timestamp.

Referenziato da aggiungi_salvataggio(), elimina_salvataggio(), e gestisci_trucchi().

6.19.2.8 stampa_salvataggi()

```
void stampa_salvataggi (
            NodoSalvataggio * testa)
```

Stampa a video l'elenco dei salvataggi disponibili.

Mostra ID, timestamp e riepilogo delle statistiche per ogni salvataggio.

Parametri

testa	Testa della lista dei salvataggi.

Definizione alla linea 130 del file salvataggio.c.

Referenzia NodoSalvataggio::dati_giocatore, NodoSalvataggio::id, Giocatore::missione_grotta, Giocatore::missione_magione, Giocatore::missione_palude, Giocatore::monete, Giocatore::numero_oggetti, NodoSalvataggio::prossimo, Giocatore::punti_vita, e NodoSalvataggio::timestamp.

Referenziato da gestisci_trucchi(), e mostra_menu_salvataggi().

6.20 salvataggio.c

Vai alla documentazione di questo file.

```
00001
00005
00006 #include "salvataggio.h"
00007 #include <stdio.h>
00008 #include <stdlib.h>
00009 #include <string.h>
00010 #include <time.h>
00011
00012 // Funzione helper per ottenere la data corrente come stringa
00013 void ottieni_data_corrente(char *buffer, size_t size)
00014 {
00015   time_t t = time(NULL);
00016   struct tm tm = *localtime(&t);
00017   // Formato: GG-MM-AAAA HH:MM:SS
00018   snprintf(buffer, size, "%02d-%02d-%04d %02d:%02d:%02d", tm.tm_mday, tm.tm_mon + 1, tm.tm_year +
      1900, tm.tm_hour, tm.tm_min, tm.tm_sec);
00019 }
00020
00021 void salva_tutto_su_file(NodoSalvataggio *testa)
00022 {
00023   FILE *f = fopen("salvataggi.bin", "wb");
00024   if (!f)
00025   {
00026     perror("Errore apertura file salvataggi");
00027     return;
00028   }
```

```
00029
00030   NodoSalvataggio *curr = testa;
00031   while (curr)
00032   {
00033     // Scriviamo ID, timestamp e dati giocatore
00034     fwrite(&curr->id, sizeof(int), 1, f);
00035     fwrite(curr->timestamp, sizeof(curr->timestamp), 1, f);
00036     fwrite(&curr->dati_giocatore, sizeof(Giocatore), 1, f);
00037
00038     // Passiamo al prossimo nodo
00039     curr = curr->prossimo;
00040   }
00041   fclose(f);
00042 }
00043
00044 void aggiungi_salvataggio(NodoSalvataggio **testa, Giocatore *g)
00045 {
00046   NodoSalvataggio *nuovo = (NodoSalvataggio *)malloc(sizeof(NodoSalvataggio));
00047   if (!nuovo)
00048   {
00049     printf("Errore di allocazione memoria per il salvataggio.\n");
00050     return;
00051   }
00052
00053   // Copia i dati del giocatore
00054   nuovo->dati_giocatore = *g;
00055   ottieni_data_corrente(nuovo->timestamp, sizeof(nuovo->timestamp));
00056   nuovo->prossimo = NULL;
00057
00058   // Calcolo ID: Massimo attuale + 1
00059   int max_id = 0;
00060   NodoSalvataggio *curr = *testa;
00061   while (curr)
00062   {
00063     if (curr->id > max_id)
00064       max_id = curr->id;
00065     curr = curr->prossimo;
00066   }
00067   nuovo->id = max_id + 1;
00068
00069   // Inserimento in testa alla lista
00070   nuovo->prossimo = *testa;
00071   *testa = nuovo;
00072
00073   // Aggiorna il file dopo l'aggiunta
00074   salva_tutto_su_file(*testa);
00075
00076   printf("Partita salvata con successo! (ID: %d)\n", nuovo->id);
00077 }
00078
00079 NodoSalvataggio *carica_salvataggio(NodoSalvataggio *testa, int id)
00080 {
00081   // Parto dalla testa e cerco il nodo con l'ID corrispondente
00082   NodoSalvataggio *curr = testa;
00083   while (curr)
00084   {
00085     if (curr->id == id)
00086     {
00087       return curr;
00088     }
00089     // Passo al prossimo nodo
00090     curr = curr->prossimo;
00091   }
00092   return NULL;
00093 }
00094
00095 void elimina_salvataggio(NodoSalvataggio **testa, int id)
00096 {
00097   NodoSalvataggio *curr = *testa;
00098   NodoSalvataggio *prev = NULL;
00099
00100   while (curr)
00101   {
00102     if (curr->id == id)
00103     {
00104       // Rimuoviamo il nodo dalla lista
00105       if (prev)
00106       {
00107         // Se non è il primo nodo, aggiorniamo il puntatore del nodo precedente
00108         prev->prossimo = curr->prossimo;
00109       }
00110       else
00111       {
00112         // Se è il primo nodo, aggiorniamo la testa della lista
00113         *testa = curr->prossimo;
00114       }
00115       free(curr);
```

```
00116
00117       // Aggiorniamo il file dopo la modifica
00118       salva_tutto_su_file(*testa);
00119
00120       printf("Salvataggio %d eliminato.\n", id);
00121       return;
00122     }
00123     // Passiamo al prossimo nodo
00124     prev = curr;
00125     curr = curr->prossimo;
00126   }
00127   printf("Salvataggio con ID %d non trovato.\n", id);
00128 }
00129
00130 void stampa_salvataggi(NodoSalvataggio *testa)
00131 {
00132   if (!testa)
00133   {
00134     printf("Nessun salvataggio disponibile.\n");
00135     return;
00136   }
00137
00138   NodoSalvataggio *curr = testa;
00139   while (curr)
00140   {
00141     // Formato richiesto: ID. DATA , P. VITA , MONETE , OGGETTI , MISSIONI COMPLETATE
00142     printf("%d. %s , %d P. VITA , %d MONETE , %d OGGETTI , %d MISSIONI COMPLETATE\n",
00143            curr->id, curr->timestamp, curr->dati_giocatore.punti_vita,
00144            curr->dati_giocatore.monete, curr->dati_giocatore.numero_oggetti,
00145            curr->dati_giocatore.missione_palude + curr->dati_giocatore.missione_magione +
    curr->dati_giocatore.missione_grotta);
00146
00147     curr = curr->prossimo;
00148   }
00149 }
00150
00151 void libera_salvataggi(NodoSalvataggio *testa)
00152 {
00153
00154   // Scorriamo la lista e liberiamo ogni nodo
00155   NodoSalvataggio *curr = testa;
00156   while (curr)
00157   {
00158     NodoSalvataggio *temp = curr;
00159     // Passiamo al prossimo nodo prima di liberare quello corrente
00160     curr = curr->prossimo;
00161     free(temp);
00162   }
00163 }
00164
00165 void carica_salvataggi_da_file(NodoSalvataggio **testa)
00166 {
00167   FILE *f = fopen("salvataggi.bin", "rb");
00168   if (!f)
00169     return; // File non esistente, nessun salvataggio
00170
00171   // Svuota lista attuale se ce ne fosse bisogno
00172   libera_salvataggi(*testa);
00173   *testa = NULL;
00174
00175   while (1)
00176   {
00177     NodoSalvataggio *nuovo = (NodoSalvataggio *)malloc(sizeof(NodoSalvataggio));
00178     if (!nuovo)
00179       break;
00180
00181     // Leggi i 3 campi (ID, timestamp, dati_giocatore)
00182
00183     if (fread(&nuovo->id, sizeof(int), 1, f) != 1) // Se non riesce a leggere l'ID, probabilmente
    siamo alla fine del file
00184     {
00185       free(nuovo); // Dealloca se fallisce la lettura
00186       break;
00187     }
00188
00189     fread(nuovo->timestamp, sizeof(nuovo->timestamp), 1, f); // Legge il timestamp
00190     fread(&nuovo->dati_giocatore, sizeof(Giocatore), 1, f);  // Legge i dati del giocatore
00191
00192     // Inserisce il nuovo nodo in fondo alla lista
00193     nuovo->prossimo = NULL;
00194     if (*testa == NULL)
00195     {
00196       // Se la lista è vuota, il nuovo nodo diventa la testa
00197       *testa = nuovo;
00198     }
00199     else
00200     {
```

```
00201       // Altrimenti, scorriamo fino alla fine della lista e aggiungiamo il nuovo nodo
00202       NodoSalvataggio *temp = *testa;
00203       while (temp->prossimo)
00204         temp = temp->prossimo;
00205       temp->prossimo = nuovo;
00206     }
00207   }
00208   fclose(f);
00209 }
```

6.21 Riferimenti per il file src/salvataggio.h

Gestione del sistema di salvataggio e caricamento.

`#include "giocatore.h"`
Grafo delle dipendenze di inclusione per salvataggio.h: Questo grafo mostra quali altri file includono direttamente o indirettamente questo file:

Strutture dati

- struct NodoSalvataggio

  Nodo di una lista concatenata per memorizzare i salvataggi.

Ridefinizioni di tipo (typedef)

- typedef struct NodoSalvataggio NodoSalvataggio

  Nodo di una lista concatenata per memorizzare i salvataggi.

Funzioni

- void aggiungi_salvataggio (NodoSalvataggio **testa, Giocatore *g)

  Aggiunge un nuovo salvataggio alla lista e al file.
- NodoSalvataggio * carica_salvataggio (NodoSalvataggio *testa, int id)

  Cerca un salvataggio nella lista tramite ID.
- void elimina_salvataggio (NodoSalvataggio **testa, int id)

  Elimina un salvataggio dalla lista e dal file.
- void stampa_salvataggi (NodoSalvataggio *testa)

  Stampa a video l'elenco dei salvataggi disponibili.
- void carica_salvataggi_da_file (NodoSalvataggio **testa)

  Carica tutti i salvataggi dal file binario all'avvio.
- void salva_tutto_su_file (NodoSalvataggio *testa)

  Salva l'intera lista dei salvataggi su file.
- void libera_salvataggi (NodoSalvataggio *testa)

  Libera la memoria allocata per la lista dei salvataggi.

6.21.1 Descrizione dettagliata

Gestione del sistema di salvataggio e caricamento.

Definizione nel file salvataggio.h.

6.21.2 Documentazione delle ridefinizioni di tipo (typedef)

6.21.2.1 NodoSalvataggio

```
typedef struct NodoSalvataggio NodoSalvataggio
```

Nodo di una lista concatenata per memorizzare i salvataggi.

6.21.3 Documentazione delle funzioni

6.21.3.1 aggiungi_salvataggio()

```
void aggiungi_salvataggio (
            NodoSalvataggio ** testa,
            Giocatore * g)
```

Aggiunge un nuovo salvataggio alla lista e al file.

Crea un nuovo nodo con i dati attuali del giocatore, genera un nuovo ID, lo aggiunge in testa alla lista e aggiorna il file di salvataggio.

Parametri

testa	Puntatore doppio alla testa della lista.
g	Puntatore ai dati del giocatore da salvare.

Definizione alla linea 44 del file salvataggio.c.

Referenzia NodoSalvataggio::dati_giocatore, NodoSalvataggio::id, ottieni_data_corrente(), NodoSalvataggio::prossimo, salva_tutto_su_file(), e NodoSalvataggio::timestamp.

Referenziato da mostra_menu_villaggio().

6.21.3.2 carica_salvataggi_da_file()

```
void carica_salvataggi_da_file (
            NodoSalvataggio ** testa)
```

Carica tutti i salvataggi dal file binario all'avvio.

Legge il file "salvataggi.bin" e ricostruisce la lista concatenata in memoria.

Parametri

testa	Puntatore doppio alla testa della lista che verrà inizializzata.

Definizione alla linea 165 del file salvataggio.c.

Referenzia NodoSalvataggio::dati_giocatore, NodoSalvataggio::id, libera_salvataggi(), NodoSalvataggio::prossimo, e NodoSalvataggio::timestamp.

Referenziato da main().

Parametri

testa	Testa della lista dei salvataggi.
id	ID del salvataggio da cercare.

Restituisce

Puntatore al nodo trovato, o NULL se non esiste.

Definizione alla linea 79 del file salvataggio.c.

Referenzia NodoSalvataggio::id, e NodoSalvataggio::prossimo.

Referenziato da gestisci_trucchi(), e mostra_menu_salvataggi().

6.21.3.4 elimina_salvataggio()

```
void elimina_salvataggio (
            NodoSalvataggio ** testa,
            int id)
```

Elimina un salvataggio dalla lista e dal file.

Rimuove il nodo corrispondente all'ID specificato e aggiorna il file di salvataggio.

Parametri

testa	Puntatore doppio alla testa della lista.
id	ID del salvataggio da eliminare.

Definizione alla linea 95 del file salvataggio.c.

Referenzia NodoSalvataggio::id, NodoSalvataggio::prossimo, e salva_tutto_su_file().

Referenziato da mostra_menu_salvataggi().

6.21.3.5 libera_salvataggi()

```
void libera_salvataggi (
            NodoSalvataggio * testa)
```

Libera la memoria allocata per la lista dei salvataggi.

Da chiamare alla chiusura del programma.

Parametri

testa	Testa della lista da deallocare.

Definizione alla linea 151 del file salvataggio.c.

Referenzia NodoSalvataggio::prossimo.

Referenziato da carica_salvataggi_da_file(), e main().

6.21.3.6 salva_tutto_su_file()

```
void salva_tutto_su_file (
            NodoSalvataggio * testa)
```

Salva l'intera lista dei salvataggi su file.

Sovrascrive il file "salvataggi.bin" con il contenuto attuale della lista.

Parametri

testa	Testa della lista dei salvataggi.

Definizione alla linea 21 del file salvataggio.c.

Referenzia NodoSalvataggio::dati_giocatore, NodoSalvataggio::id, NodoSalvataggio::prossimo, e NodoSalvataggio::timestamp.

Referenziato da aggiungi_salvataggio(), elimina_salvataggio(), e gestisci_trucchi().

6.21.3.7 stampa_salvataggi()

```
void stampa_salvataggi (
            NodoSalvataggio * testa)
```

Stampa a video l'elenco dei salvataggi disponibili.

Mostra ID, timestamp e riepilogo delle statistiche per ogni salvataggio.

Parametri

testa	Testa della lista dei salvataggi.

Definizione alla linea 130 del file salvataggio.c.

Referenzia NodoSalvataggio::dati_giocatore, NodoSalvataggio::id, Giocatore::missione_grotta, Giocatore::missione_magione, Giocatore::missione_palude, Giocatore::monete, Giocatore::numero_oggetti, NodoSalvataggio::prossimo, Giocatore::punti_vita, e NodoSalvataggio::timestamp.

Referenziato da gestisci_trucchi(), e mostra_menu_salvataggi().

6.22 salvataggio.h

Vai alla documentazione di questo file.

```
00001
00005
00006 #ifndef SALVATAGGIO_H
00007 #define SALVATAGGIO_H
00008
00009 #include "giocatore.h"
00010
00014 typedef struct NodoSalvataggio
00015 {
00016   int id;
00017   char timestamp[32];
00018   Giocatore dati_giocatore;
00019   struct NodoSalvataggio *prossimo;
00020 } NodoSalvataggio;
00021
00031 void aggiungi_salvataggio(NodoSalvataggio **testa, Giocatore *g);
00032
00040 NodoSalvataggio *carica_salvataggio(NodoSalvataggio *testa, int id);
00041
00050 void elimina_salvataggio(NodoSalvataggio **testa, int id);
00051
00059 void stampa_salvataggi(NodoSalvataggio *testa);
00060
00068 void carica_salvataggi_da_file(NodoSalvataggio **testa);
00069
00077 void salva_tutto_su_file(NodoSalvataggio *testa);
00078
00086 void libera_salvataggi(NodoSalvataggio *testa);
00087
00088 #endif
```

6.23 Riferimenti per il file src/stanza.h

Definizione della struttura e dei tipi di Stanza.

Questo grafo mostra quali altri file includono direttamente o indirettamente questo file:

Strutture dati

- struct Stanza

  Struttura che rappresenta una stanza del dungeon.

Tipi enumerati (enum)

- enum TipoStanza { STANZA_NESSUNA , STANZA_NEMICO , STANZA_TRAPPOLA , STANZA_VUOTA }

  Enum che rappresenta i diversi tipi di stanza.

6.23.1 Descrizione dettagliata

Definizione della struttura e dei tipi di Stanza.

Definizione nel file stanza.h.

6.23.2 Documentazione dei tipi enumerati

6.23.2.1 TipoStanza

```
enum TipoStanza
```

Valori del tipo enumerato

STANZA_NESSUNA	Stanza non definita
STANZA_NEMICO	Stanza con un nemico da affrontare
STANZA_TRAPPOLA	Stanza con una trappola da superare
STANZA_VUOTA	Stanza vuota senza eventi

Definizione alla linea 12 del file stanza.h.

6.24 stanza.h

Vai alla documentazione di questo file.

```
00001
00005
00006 #ifndef STANZA_H
00007 #define STANZA_H
00008
00012 typedef enum
00013 {
00014   STANZA_NESSUNA,
00015   STANZA_NEMICO,
00016   STANZA_TRAPPOLA,
00017   STANZA_VUOTA
00018 } TipoStanza;
00019
00023 typedef struct
00024 {
00025   char nome[32];
00026   TipoStanza tipo;
00027   int danno;
00028   int colpo_fatale;
00029   int ricompensa_monete;
00030   int is_boss;
00031 } Stanza;
00032
00033 #endif
```

6.25 Riferimenti per il file src/utilita.c

Implementazione delle funzioni di utilità.

```
#include "utilita.h"
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <string.h>
```

Grafo delle dipendenze di inclusione per utilita.c:

Funzioni

- int lancia_dado (int facce)

  Lancia un dado con un specificato numero di facce.
- int padovan (int n)

  Calcola l'n-esimo numero della sequenza di Padovan.
- int controlla_padovan (int num)

  Verifica se un numero appartiene alla sequenza di Padovan.

- void pulisci_schermo ()

  Pulisce lo schermo del terminale.
- char leggi_input_char ()

  Legge un singolo carattere da input.
- int leggi_intero ()

  Legge un numero intero da input.
- int valuta_vittoria_morra (int m1, int m2)

  Valuta il vincitore nella Morra Cinese.

6.25.1 Descrizione dettagliata

Implementazione delle funzioni di utilità.

Definizione nel file utilita.c.

6.25.2 Documentazione delle funzioni

6.25.2.1 controlla_padovan()

```
int controlla_padovan (
            int num)
```

Verifica se un numero appartiene alla sequenza di Padovan.

Controlla i primi 25 numeri della sequenza.

Parametri

num	Il numero da verificare.

Restituisce

1 se il numero appartiene alla sequenza, 0 altrimenti.

Definizione alla linea 25 del file utilita.c.

Referenzia padovan().

Referenziato da inizia_combattimento().

6.25.2.2 lancia_dado()

```
int lancia_dado (
            int facce)
```

Lancia un dado con un specificato numero di facce.

Parametri

facce	Il numero di facce del dado.

Restituisce

Un numero casuale tra 1 e facce.

Definizione alla linea 12 del file utilita.c.

Referenziato da combattimento_boss_finale(), esegui_missione(), genera_stanza_missione(), inizia_combattimento(), e mostra_negozio().

6.25.2.3 leggi_input_char()

```
char leggi_input_char ()
```

Legge un singolo carattere da input.

Pulisce il buffer e gestisce l'input dell'utente, restituendo il carattere in minuscolo.

Restituisce

Il carattere letto, oppure 0 in caso di input vuoto o errore.

Definizione alla linea 50 del file utilita.c.

Referenziato da mostra_menu_principale(), mostra_menu_salvataggi(), e mostra_menu_villaggio().

6.25.2.4 leggi_intero()

```
int leggi_intero ()
```

Legge un numero intero da input.

Gestisce la validazione dell'input richiedendo l'inserimento finché non viene fornito un intero valido.

Restituisce

Il numero intero letto.

Definizione alla linea 76 del file utilita.c.

Referenziato da combattimento_boss_finale(), esegui_missione(), gestisci_trucchi(), mostra_menu_missione(), mostra_menu_salvataggi(), mostra_menu_villaggio(), e mostra_negozio().

6.25.2.5 padovan()

```
int padovan (
            int n)
```


Calcola l'n-esimo numero della sequenza di Padovan.

Parametri

n	L'indice della sequenza.

Restituisce

L'n-esimo numero di Padovan.

Definizione alla linea 16 del file utilita.c.

Referenzia padovan().

Referenziato da controlla_padovan(), e padovan().

6.25.2.6 pulisci_schermo()

```
void pulisci_schermo ()
```

Pulisce lo schermo del terminale.

Supporta sia sistemi Windows che UNIX-like.

Definizione alla linea 40 del file utilita.c.

Referenziato da esegui_missione(), gestisci_trucchi(), mostra_menu_missione(), mostra_menu_principale(), mostra_menu_villaggio(), e mostra_negozio().

6.25.2.7 valuta_vittoria_morra()

```
int valuta_vittoria_morra (
            int mossa1,
            int mossa2)
```

Valuta il vincitore nella Morra Cinese.

Le regole sono:

- Scudo (1) batte Spada (3)
- Magia (2) batte Scudo (1)
- Spada (3) batte Magia (2)

Parametri

mossa1	La mossa del primo giocatore.
mossa2	La mossa del secondo giocatore (boss).

Restituisce

0 per pareggio, 1 se vince mossa1, 2 se vince mossa2.

Definizione alla linea 91 del file utilita.c.

Referenziato da combattimento_boss_finale().

6.26 utilita.c

Vai alla documentazione di questo file.

```
00001
00005
00006 #include "utilita.h"
00007 #include <stdio.h>
00008 #include <stdlib.h>
00009 #include <ctype.h>
00010 #include <string.h>
00011
00012 int lancia_dado(int facce) { return (rand() % facce) + 1; }
00013
00014 // P(n) = P(n-2) + P(n-3)
00015 // P(0)=1, P(1)=1, P(2)=1
00016 int padovan(int n)
00017 {
00018   if (n < 0)
00019     return 0;
00020   if (n <= 2)
00021     return 1;
00022   return padovan(n - 2) + padovan(n - 3);
00023 }
00024
00025 int controlla_padovan(int num)
00026 {
00027   if (num < 1)
00028     return 0;
00029   for (int i = 0; i < 25; i++)
00030   {
00031     int p = padovan(i);
00032     if (p == num)
00033       return 1;
00034     if (p > num)
00035       return 0;
00036   }
00037   return 0;
00038 }
00039
00040 void pulisci_schermo()
00041 {
00042 #ifdef _WIN32
00043   system("cls");
00044 #else
00045   system("clear");
00046 #endif
00047 }
00048
00049 // Legge un singolo carattere dall'input dell'utente e lo restituisce in minuscolo
00050 char leggi_input_char()
00051 {
00052   char b[100]; // Buffer per memorizzare l'input (100 caratteri max)
00053
00054   // Legge una linea intera da stdin in modo sicuro
00055   if (fgets(b, sizeof(b), stdin))
00056   {
00057     // Rimuove il carattere di newline inserito automaticamente da fgets
00058     b[strcspn(b, "\n")] = 0;
00059
00060     // Se l'utente ha premuto solo invio (input vuoto), restituisce 0
00061     if (strlen(b) == 0)
00062       return 0;
00063
00064     // Se l'utente ha inserito solo uno spazio, lo restituisce
00065     if (strcmp(b, " ") == 0)
00066       return ' ';
00067
00068     // Restituisce il primo carattere convertito a minuscolo
00069     return tolower(b[0]);
00070   }
00071
00072   // Se fgets fallisce, restituisce 0
00073   return 0;
00074 }
00075
00076 int leggi_intero()
00077 {
00078   int valore;
00079   while (scanf("%d", &valore) != 1) // Continua a chiedere finché non viene inserito un intero valido
00080   {
00081     while (getchar() != '\n')
00082       ; // Pulisci buffer
00083     printf("Input non valido. Inserisci un numero: ");
00084   }
00085   while (getchar() != '\n')
```

```
00086     ; // Pulisci fine riga
00087   return valore;
00088 }
00089
00090 // 1: Scudo, 2: Magia, 3: Spada
00091 int valuta_vittoria_morra(int m1, int m2)
00092 {
00093   if (m1 == m2)
00094     return 0;
00095
00096   // Scudo(1) > Spada(3)
00097   // Magia(2) > Scudo(1)
00098   // Spada(3) > Magia(2)
00099   if ((m1 == 1 && m2 == 3) || (m1 == 2 && m2 == 1) || (m1 == 3 && m2 == 2))
00100   {
00101     return 1;
00102   }
00103   return 2;
00104 }
```

6.27 Riferimenti per il file src/utilita.h

Funzioni di utilità generale per il gioco.

Questo grafo mostra quali altri file includono direttamente o indirettamente questo file:

Funzioni

- int lancia_dado (int facce)

  Lancia un dado con un specificato numero di facce.
- int padovan (int n)

  Calcola l'n-esimo numero della sequenza di Padovan.
- int controlla_padovan (int num)

  Verifica se un numero appartiene alla sequenza di Padovan.
- void pulisci_schermo ()

  Pulisce lo schermo del terminale.
- char leggi_input_char ()

  Legge un singolo carattere da input.
- int leggi_intero ()

  Legge un numero intero da input.
- int valuta_vittoria_morra (int mossa1, int mossa2)

  Valuta il vincitore nella Morra Cinese.

6.27.1 Descrizione dettagliata

Funzioni di utilità generale per il gioco.

Contiene funzioni per la generazione di numeri casuali, calcoli matematici (sequenza di Padovan), gestione input/↩ output e logica di base per la Morra Cinese.

Definizione nel file utilita.h.

6.27.2 Documentazione delle funzioni

6.27.2.1 controlla_padovan()


```
int controlla_padovan (
            int num)
```

Parametri

num	Il numero da verificare.

Restituisce

1 se il numero appartiene alla sequenza, 0 altrimenti.

Definizione alla linea 25 del file utilita.c.

Referenzia padovan().

Referenziato da inizia_combattimento().

6.27.2.2 lancia_dado()

```
int lancia_dado (
            int facce)
```

Lancia un dado con un specificato numero di facce.

Parametri

facce	Il numero di facce del dado.

Restituisce

Un numero casuale tra 1 e facce.

Definizione alla linea 12 del file utilita.c.

Referenziato da combattimento_boss_finale(), esegui_missione(), genera_stanza_missione(), inizia_combattimento(), e mostra_negozio().

6.27.2.3 leggi_input_char()

```
char leggi_input_char ()
```

Legge un singolo carattere da input.

Pulisce il buffer e gestisce l'input dell'utente, restituendo il carattere in minuscolo.

Restituisce

Il carattere letto, oppure 0 in caso di input vuoto o errore.

Definizione alla linea 50 del file utilita.c.

Referenziato da mostra_menu_principale(), mostra_menu_salvataggi(), e mostra_menu_villaggio().

6.27.2.4 leggi_intero()

```
int leggi_intero ()
```

Legge un numero intero da input.

Gestisce la validazione dell'input richiedendo l'inserimento finché non viene fornito un intero valido.

Restituisce

Il numero intero letto.

Definizione alla linea 76 del file utilita.c.

Referenziato da combattimento_boss_finale(), esegui_missione(), gestisci_trucchi(), mostra_menu_missione(), mostra_menu_salvataggi(), mostra_menu_villaggio(), e mostra_negozio().

6.27.2.5 padovan()

```
int padovan (
            int n)
```

Calcola l'n-esimo numero della sequenza di Padovan.

La sequenza è definita come P(n) = P(n-2) + P(n-3) con P(0)=1, P(1)=1, P(2)=1.

Parametri

n	L'indice della sequenza.

Restituisce

L'n-esimo numero di Padovan.

Definizione alla linea 16 del file utilita.c.

Referenzia padovan().

Referenziato da controlla_padovan(), e padovan().

6.27.2.6 pulisci_schermo()

```
void pulisci_schermo ()
```

Pulisce lo schermo del terminale.

Supporta sia sistemi Windows che UNIX-like.

Definizione alla linea 40 del file utilita.c.

Referenziato da esegui_missione(), gestisci_trucchi(), mostra_menu_missione(), mostra_menu_principale(), mostra_menu_villaggio(), e mostra_negozio().

6.27.2.7 valuta_vittoria_morra()

```
int valuta_vittoria_morra (
            int mossa1,
            int mossa2)
```

Valuta il vincitore nella Morra Cinese.

Le regole sono:

- Scudo (1) batte Spada (3)
- Magia (2) batte Scudo (1)
- Spada (3) batte Magia (2)

Parametri

mossa1	La mossa del primo giocatore.
mossa2	La mossa del secondo giocatore (boss).

Restituisce

0 per pareggio, 1 se vince mossa1, 2 se vince mossa2.

Definizione alla linea 91 del file utilita.c.

Referenziato da combattimento_boss_finale().

6.28 utilita.h

Vai alla documentazione di questo file.

```
00001
00010
00011 #ifndef UTILITA_H
00012 #define UTILITA_H
00013
00019 int lancia_dado(int facce);
00020
00021 // Funzioni matematiche / logiche
00022
00032 int padovan(int n);
00033
00042 int controlla_padovan(int num);
00043
00044 // Funzioni input/output
00045
00051 void pulisci_schermo();
00052
00061 char leggi_input_char();
00062
00071 int leggi_intero();
00072
00073 // Logica Morra Cinese (1: Scudo, 2: Magia, 3: Spada)
00074
00087 int valuta_vittoria_morra(int mossa1, int mossa2);
00088
00089 #endif
```