My Project

Generato da Doxygen 1.13.2

Capitolo 1

prg00819

Capitolo 2

Indice della gerarchia

2.1 Gerarchia delle classi

Questo elenco di ereditarietà è ordinato approssimativamente, ma non completamente, in ordine alfabetico:

Capitolo 3

Indice dei tipi composti

3.1 Elenco dei tipi composti

Queste sono le classi, le struct, le union e le interfacce con una loro breve descrizione:

Capitolo 4

Indice dei file

4.1 Elenco dei file

Questo è un elenco dei file documentati con una loro breve descrizione:

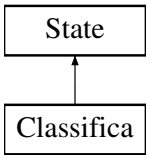
Capitolo 5

Documentazione delle classi

5.1 Riferimenti per la classe Classifica

`#include <classifica.hpp>`

Diagramma delle classi per Classifica



Membri pubblici

- Classifica ()

  Costruttore dello stato.
- void display ()

  Stampa la classifica nel terminale.
- int update (int input)

  Fa interagire l'utente con lo stato di giocos.
- player **gameOverScreen** (int points)

  ???
- void get_new_name (char name[ ])

  Fa immettere all'utente il suo nome per la classifica.
- void addPlayer ()

  Aggiunge un nuovo giocatore nella classifica.
- void insertPlayer (char name[ ], int points)

  Inserisce un giocatore nellla lista concatenata.

Membri pubblici ereditati da State

- **State** ()

  Costruttore della classe.
- void setDone (int d)

  Comunica che lo stato ha finito.
- int isDone ()

  Controlla se lo staato ha finito.
- void setNext (state n)

  Decide quale sarà il prossimo stato.
- state getNext ()

  Fa sapere il prossimo stato di gioco.
- void setPrev (state prev)

  Decide lo stato precedente.
- void Quit ()

  Esce definitivamente dal gioco.
- int getQuit ()

  Fa sapere se bisogna uscire dal gioco.

Altri membri ereditati

Attributi protetti ereditati da State

- int **done** = 0
- int **quit** = 0
- state **next**
- state **previous**

5.1.1 Descrizione dettagliata

Stato per la classifica di gioco

5.1.2 Documentazione dei costruttori e dei distruttori

5.1.2.1 Classifica()

```
Classifica::Classifica ()
```

Costruttore dello stato.

Costruttore della classe

5.1.3 Documentazione delle funzioni membro

5.1.3.1 addPlayer()

```
void Classifica::addPlayer ()
```

Aggiunge un nuovo giocatore nella classifica.

Aggiunge un nuovo giocatore in classifica

5.1.3.2 display()

```
void Classifica::display ()
```

Stampa la classifica nel terminale.

Stampa la classifica nel terminale

5.1.3.3 get_new_name()

```
void Classifica::get_new_name (
            char name[])
```

Fa immettere all'utente il suo nome per la classifica.

Fa immetere all'utente il suo nome nel terminale

Parametri

new_name	il nuovo nome utente

5.1.3.4 insertPlayer()

```
void Classifica::insertPlayer (
            char name[],
            int points)
```

Inserisce un giocatore nellla lista concatenata.

Inserisce un utente nella lista concatenata

5.1.3.5 update()

```
int Classifica::update (
            int input)  [virtual]
```

Fa interagire l'utente con lo stato di giocos.

Funzione in cui l'utente può interagire con lo stato

Restituisce

il tasto premuto dall'utente

Implementa State.

La documentazione per questa classe è stata generata a partire dai seguenti file:

- include/states/classifica.hpp
- include/states/classifica.cpp

5.2 Riferimenti per la struct f

`#include <screen.hpp>`

Attributi pubblici

- char * **str** [50]
- int **max_len**

5.2.1 Descrizione dettagliata

Lista concatenata per scrivere efficientemente le liste di stringhe in una finestra

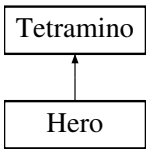
La documentazione per questa struct è stata generata a partire dal seguente file:

- include/states/screen.hpp

5.3 Riferimenti per la classe Hero

`#include <hero.hpp>`

Diagramma delle classi per Hero



Membri pubblici

- Hero (World world)

  Costruttore del tetramino.
- void print_frame ()

  Stampa il tetramino nel suo frame attuale.
- void safe_move (int dir)

  Muove il tetramino senza farlo uscire dal mondo.
- void rotate ()

  Ruota il tetramino di mezzo angolo giro.
- int side_collisions ()

  Controlla che non ci siano ostacoli ai lati.

Membri pubblici ereditati da Tetramino

- Tetramino (World world, int w, int h)

  Costruttore della classe.
- void print (int shape)

  Stampa il tetramino nel terminale.
- void move (int dir)

  Muove in tetramino a destra o a sinistra.
- int falling ()

  Simulazione della caduta del tetramino.
- void dies ()

  Morte del tetramino.
- int check_collision ()

  Controlla le collisioni del tetramino.
- void getclout (int row, char *buffer)

  Fa sapere il contenuto di una riga del tetramino.

Attributi protetti

- long int frames [FRLEN]
- int **current** =0

Attributi protetti ereditati da Tetramino

- int **SCRW**
- int **SCRH**
- int **XOFF**
- int **WIDTH**
- int **HEIGHT**
- int **STARTX** =20
- int **STARTY** =1
- int **x**
- int **y**
- WINDOW * **base**

5.3.1 Descrizione dettagliata

Tetramino verticale, chiamato con il nome originale

5.3.2 Documentazione dei costruttori e dei distruttori

5.3.2.1 Hero()

```
Hero::Hero (
            World world)
```

Costruttore del tetramino.

Costruttore della classe

Parametri

world	il mondo in cui esisterà il tetramino

5.3.3 Documentazione delle funzioni membro

5.3.3.1 print_frame()

```
void Hero::print_frame ()  [virtual]
```

Stampa il tetramino nel suo frame attuale.

Stampa il tetramino nel suo frame attuale

Implementa Tetramino.

5.3.3.2 rotate()

```
void Hero::rotate ()  [virtual]
```

Ruota il tetramino di mezzo angolo giro.

Ruota il tetramino di mezzo angolo giro

Implementa Tetramino.

5.3.3.3 safe_move()

```
void Hero::safe_move (
            int dir)  [virtual]
```

Muove il tetramino senza farlo uscire dal mondo.

Muove il tetramino in modo che non strabordi fuori dal mondo

Parametri

dir	positiva se a destra negativa altrimenti

Implementa Tetramino.

5.3.3.4 side_collisions()

```
int Hero::side_collisions ()
```

Controlla che non ci siano ostacoli ai lati.

Controllo se ci sono ostacoli ai lati del tetramino

5.3.4 Documentazione dei membri dato

5.3.4.1 frames

```
long int Hero::frames[FRLEN]  [protected]
```

Valore iniziale:

```
= {
            61440, 8738

        }
```

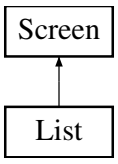
La documentazione per questa classe è stata generata a partire dai seguenti file:

- include/tetramini/hero.hpp
- include/tetramini/hero.cpp

5.4 Riferimenti per la classe List

```
#include <screen.hpp>
```

Diagramma delle classi per List



Membri pubblici

- **List** ()

  Costruttore della classe.
- void init (char text[ ], char delimiter, int r_s, int txt_xoff, int txt_yoff, char borders[ ])

  Inizializza la finestra.
- void compute_sizes (int ∗width, int ∗height)

  Aggiorna larghezza e altezza.
- void update_list (char new_text[ ])

  Aggiorna la lista di stringhe da stampare.
- void show_list ()

  Stampa la finestra e la lista di stringhe.
- int get_len ()

  Fa sapere la lunghezza della lista di stringhe.

Membri pubblici ereditati da Screen

- **Screen** ()

  Costruttore della classe.
- void init (int w, int h, int stx, int sty, char borders[ ])

  Inizializza la finestra.
- void setInCenter ()

  Centra la finestra nel terminale.
- void show ()

  Stampa la finestra.
- void destroy ()

  Cancella la finestra.

Altri membri ereditati

Attributi pubblici ereditati da Screen

- WINDOW * **win**

Attributi protetti ereditati da Screen

- int **WIDTH**
- int **HEIGHT**
- int **STARTY**
- int **STARTX**
- char **BORDERS** [8]

5.4.1 Descrizione dettagliata

Classe figlia per le finestre che stampano una lista di stringhe

5.4.2 Documentazione delle funzioni membro

5.4.2.1 compute_sizes()

```
void List::compute_sizes (
            int * width,
            int * height)
```

Aggiorna larghezza e altezza.

In base alla lista di stringhe attuali calcola una nuova larghezza e altezza della finestra

Parametri

out	*width*	la nuova larghezza
out	*height*	la nuova altezza

5.4.2.2 get_len()

```
void List::get_len ()
```

Fa sapere la lunghezza della lista di stringhe.

Fa sapere la lunghezza della lista di stringhe

Restituisce

la lunghezza della lista

5.4.2.3 init()

```
void List::init (
            char text[],
            char delimiter,
            int r_s,
            int txt_xoff,
            int txt_yoff,
            char borders[])
```

Inizializza la finestra.

Inizializza tutte le proprietà principali della classe

Parametri

text[]	la lista di stringhe iniziali da stampare
delimiter	il carattere che dice quando una stringa è finita
r_s	quante righe vuote tra una stringa e l'altra
txt_xoff	quanto spazio tra i bordi verticali
txt_yoff	quanto spazio tra i bordi orizzontali
borders	stringa di 8 caratteri che sono i bordi della finestra

5.4.2.4 show_list()

```
void List::show_list ()
```

Stampa la finestra e la lista di stringhe.

Stampa la finestra e la lista di stringhe

5.4.2.5 update_list()

```
void List::update_list (
            char new_text[])
```

Aggiorna la lista di stringhe da stampare.

Aggiorna la lista di stringhe da stampare

Parametri

new_text[ ]	la nuova lista di stringhe

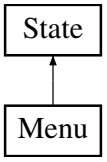
La documentazione per questa classe è stata generata a partire dai seguenti file:

- include/states/screen.hpp
- include/states/screen.cpp

5.5 Riferimenti per la classe Menu

`#include <menu.hpp>`

Diagramma delle classi per Menu



Membri pubblici

- Menu ()
  Costruttore dello stato.
- void draw ()
  Stampa le opzioni con il cursore.
- int update (int input)
  Fa interagire il giocatore con le opzioni.

Membri pubblici ereditati da State

- **State** ()
  Costruttore della classe.
- void setDone (int d)
  Comunica che lo stato ha finito.
- int isDone ()
  Controlla se lo staato ha finito.
- void setNext (state n)
  Decide quale sarà il prossimo stato.
- state getNext ()
  Fa sapere il prossimo stato di gioco.
- void setPrev (state prev)
  Decide lo stato precedente.
- void Quit ()
  Esce definitivamente dal gioco.
- int getQuit ()
  Fa sapere se bisogna uscire dal gioco.

Altri membri ereditati

Attributi protetti ereditati da State

- int **done** = 0
- int **quit** = 0
- state **next**
- state **previous**

5.5.1 Descrizione dettagliata

Stato di gioco inizile, in cui l'utente deve scegliere tra iniziare una nuova partita e il vedere la classifica

5.5.2 Documentazione dei costruttori e dei distruttori

5.5.2.1 Menu()

```
Menu::Menu ()
```

Costruttore dello stato.

Costruttore della classe

5.5.3 Documentazione delle funzioni membro

5.5.3.1 draw()

```
void Menu::draw ()
```

Stampa le opzioni con il cursore.

Stampa la finestra con il cursore e le opzioni

5.5.3.2 update()

```
int Menu::update (
            int input)  [virtual]
```

Fa interagire il giocatore con le opzioni.

Funzione che fa interagire l'utente con lo stato in base all'input

In questo caso la funzione serve a far muovere l'utente con il cursore e farlo scegliere un'opzione

Parametri

input	il tasto che preme l'utente

Implementa State.

La documentazione per questa classe è stata generata a partire dai seguenti file:

- include/states/menu.hpp
- include/states/menu.cpp

5.6 Riferimenti per la struct node

`#include <world.hpp>`

Attributi pubblici

- node * **next**
- int **val**
- int **row_blocks**

5.6.1 Descrizione dettagliata

Lista concatenata rappresenta un singolo blocco di tetris

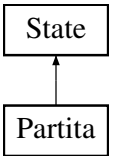
La documentazione per questa struct è stata generata a partire dal seguente file:

- include/states/world.hpp

5.7 Riferimenti per la classe Partita

`#include <gameplay.hpp>`

Diagramma delle classi per Partita



Membri pubblici

- Partita ()
  Costruttore dello stato.
- int update (int input)
  Fa interagire il giocatore con il gioco.
- int gameplay (int input)
  Fa giocare l'utente a tetris.
- void gameOver ()
  Aggiorna la classifica in caso di game over.

Membri pubblici ereditati da State

- **State** ()

  Costruttore della classe.
- void setDone (int d)

  Comunica che lo stato ha finito.
- int isDone ()

  Controlla se lo staato ha finito.
- void setNext (state n)

  Decide quale sarà il prossimo stato.
- state getNext ()

  Fa sapere il prossimo stato di gioco.
- void setPrev (state prev)

  Decide lo stato precedente.
- void Quit ()

  Esce definitivamente dal gioco.
- int getQuit ()

  Fa sapere se bisogna uscire dal gioco.

Altri membri ereditati

Attributi protetti ereditati da State

- int **done** = 0
- int **quit** = 0
- state **next**
- state **previous**

5.7.1 Descrizione dettagliata

Stato di gioco in cui si gestisce la partita di tetris

5.7.2 Documentazione dei costruttori e dei distruttori

5.7.2.1 Partita()

```
Partita::Partita ()
```

Costruttore dello stato.

Costruttore della classe

5.7.3 Documentazione delle funzioni membro

5.7.3.1 gameOver()

```
void Partita::gameOver ()
```

Aggiorna la classifica in caso di game over.

In caso di game over bisogna aggiornare la classifica

5.7.3.2 gameplay()

```
int Partita::gameplay (
            int input)
```

Fa giocare l'utente a tetris.

Fa procedere la partita e muove il tetramino secondo le regole del gioco

Parametri

input	il tasto che preme l'utente

Restituisce

se la partita è finita o no

5.7.3.3 update()

```
int Partita::update (
            int input)  [virtual]
```

Fa interagire il giocatore con il gioco.

Funzione per far interagire l'utente con lo stato di gioco

Parametri

input	il tasto che preme l'utente

Implementa State.

La documentazione per questa classe è stata generata a partire dai seguenti file:

- include/states/gameplay.hpp
- include/states/gameplay.cpp

5.8 Riferimenti per la struct pnode

Attributi pubblici

- pnode * **next**
- char **all** [100]
- char **name** [50]
- int **points**

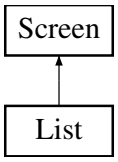
La documentazione per questa struct è stata generata a partire dal seguente file:

- include/states/classifica.hpp

5.9 Riferimenti per la classe Screen

`#include <screen.hpp>`

Diagramma delle classi per Screen



Membri pubblici

- **Screen** ()

  Costruttore della classe.
- void init (int w, int h, int stx, int sty, char borders[ ])

  Inizializza la finestra.
- void setInCenter ()

  Centra la finestra nel terminale.
- void show ()

  Stampa la finestra.
- void destroy ()

  Cancella la finestra.

Attributi pubblici

- WINDOW ∗ **win**

Attributi protetti

- int **WIDTH**
- int **HEIGHT**
- int **STARTY**
- int **STARTX**
- char **BORDERS** [8]

5.9.1 Descrizione dettagliata

Raggruppa tutte le funzioni utili per gestire una finestra ncurses

5.9.2 Documentazione delle funzioni membro

5.9.2.1 destroy()

`void Screen::destroy ()`

Cancella la finestra.

Cancella e distrugge la finestra

5.9.2.2 init()

```
void Screen::init (
            int w,
            int h,
            int stx,
            int sty,
            char borders[])
```

Inizializza la finestra.

Inizializza la finestra principale

Parametri

w	la larghezza della finestra
h	l'alezza della finestra
stx	l'ascissa di partenza della finestra
sty	l'ordinata di partenza della finestra
borders	stringa di 8 caratteri che sono i bordi della finestra

5.9.2.3 setInCenter()

```
void Screen::setInCenter ()
```

Centra la finestra nel terminale.

Per centra la finestra nel terminale

5.9.2.4 show()

```
void Screen::show ()
```

Stampa la finestra.

Stampa la finestra, con i suoi bordi e il suo contenuto

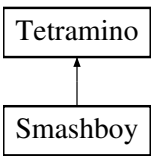
La documentazione per questa classe è stata generata a partire dai seguenti file:

- include/states/screen.hpp
- include/states/screen.cpp

5.10 Riferimenti per la classe Smashboy

```
#include <smashboy.hpp>
```

Diagramma delle classi per Smashboy



Membri pubblici

- Smashboy (World world)

  Costruttore del tetramino.
- void print_frame ()

  Stampa il tetramino nel suo frame attuale.
- void safe_move (int dir)

  Muove il tetramino senza farlo uscire dal mondo.
- void rotate ()

  Ruota il tetramino di zero gradi.

Membri pubblici ereditati da Tetramino

- Tetramino (World world, int w, int h)

  Costruttore della classe.
- void print (int shape)

  Stampa il tetramino nel terminale.
- void move (int dir)

  Muove in tetramino a destra o a sinistra.
- int falling ()

  Simulazione della caduta del tetramino.
- void dies ()

  Morte del tetramino.
- int check_collision ()

  Controlla le collisioni del tetramino.
- void getclout (int row, char ∗buffer)

  Fa sapere il contenuto di una riga del tetramino.

Attributi protetti

- int **frame** = 15

Attributi protetti ereditati da Tetramino

- int **SCRW**
- int **SCRH**
- int **XOFF**
- int **WIDTH**
- int **HEIGHT**
- int **STARTX** =20
- int **STARTY** =1
- int **x**
- int **y**
- WINDOW ∗ **base**

5.10.1 Descrizione dettagliata

Tetramino di forma quadrata, chiamato con il nome originale

5.10.2 Documentazione dei costruttori e dei distruttori

5.10.2.1 Smashboy()

```
Smashboy::Smashboy (
            World world)
```

Costruttore del tetramino.

Costruttore della classe

Parametri

world	il mondo in cui esisterà il tetramino

5.10.3 Documentazione delle funzioni membro

5.10.3.1 print_frame()

```
void Smashboy::print_frame ()  [virtual]
```

Stampa il tetramino nel suo frame attuale.

Strampa il tetraminino nel suo frame attuale

Implementa Tetramino.

5.10.3.2 rotate()

```
void Smashboy::rotate ()  [inline], [virtual]
```

Ruota il tetramino di zero gradi.

Implementa Tetramino.

5.10.3.3 safe_move()

```
void Smashboy::safe_move (
            int dir)  [virtual]
```

Muove il tetramino senza farlo uscire dal mondo.

Muove il tetramino senza farlo strabordare fuori dal mondo

Parametri

dir	positiva se a destra negativa altrimenti

Implementa Tetramino.

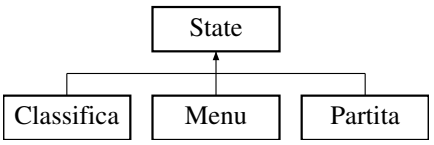
La documentazione per questa classe è stata generata a partire dai seguenti file:

- include/tetramini/smashboy.hpp
- include/tetramini/smashboy.cpp

5.11 Riferimenti per la classe State

`#include <state.hpp>`

Diagramma delle classi per State



Membri pubblici

- **State** ()
  Costruttore della classe.
- void setDone (int d)
  Comunica che lo stato ha finito.
- int isDone ()
  Controlla se lo staato ha finito.
- void setNext (state n)
  Decide quale sarà il prossimo stato.
- state getNext ()
  Fa sapere il prossimo stato di gioco.
- void setPrev (state prev)
  Decide lo stato precedente.
- void Quit ()
  Esce definitivamente dal gioco.
- int getQuit ()
  Fa sapere se bisogna uscire dal gioco.
- virtual int update (int input)=0
  Funzione astratta.

Attributi protetti

- int **done** = 0
- int **quit** = 0
- state **next**
- state **previous**

5.11.1 Descrizione dettagliata

Classe astratta dello stato di gioco. Ogni stato ha una classe concreta apposta in un suo file.

5.11.2 Documentazione delle funzioni membro

5.11.2.1 getNext()

```
state State::getNext ()
```

Fa sapere il prossimo stato di gioco.

Per vedere il prossimo stato di gioco

Restituisce

il prossimo stato

5.11.2.2 getQuit()

```
int State::getQuit ()
```

Fa sapere se bisogna uscire dal gioco.

Per controllare se uscire dal gioco

Restituisce

0 se si resta 1 altrimenti

5.11.2.3 isDone()

```
int State::isDone ()
```

Controlla se lo staato ha finito.

Per controllare se lo stato è finito

Restituisce

se il valore è 0 o 1

5.11.2.4 Quit()

```
void State::Quit ()
```

Esce definitivamente dal gioco.

Per uscire definitivamente dal gioco

5.11.2.5 setDone()

```
void State::setDone (
            int d)
```

Comunica che lo stato ha finito.

Comunica che lo stato è a fine ciclo e che bisogna passare al prossimo

Parametri

d	1 se abbiamo finito 0 altrimenti

5.11.2.6 setNext()

```
void State::setNext (
            state n)
```

Decide quale sarà il prossimo stato.

Per poter decidere il prossimo stato di gioco

Parametri

n	il prossimo stato

5.11.2.7 setPrev()

```
void State::setPrev (
            state prev)
```

Decide lo stato precedente.

Per decidere qual è lo stato precedente

Parametri

lo	stato precedente

5.11.2.8 update()

```
virtual int State::update (
            int input)  [pure virtual]
```

Funzione astratta.

Implementato in Classifica, Menu, e Partita.

La documentazione per questa classe è stata generata a partire dai seguenti file:

- include/state_machine/state.hpp
- include/state_machine/state.cpp

5.12 Riferimenti per la classe StateMachine

```
#include <state_machine.hpp>
```

Membri pubblici

- StateMachine (state start)
  Costruttore della classe.
- void init_ncurses ()
  Inizializza le funzioni di ncurses.
- void flip ()
  Cambia stato di gioco.
- void update (int input)
  Aggiorna lo stato attuale.
- void game_loop ()
  Ciclo principale del gioco.

Attributi protetti

- int **done** = 0
- state **current_state**
- State ∗ **states** [STATES_LEN]
- State ∗ **current**

5.12.1 Descrizione dettagliata

Implentazione di una macchina a stati minimalista

Controlla gli stati del gioco e ne facilita il cambio al verificarsi di certi eventi. Permette gestire meglio le interazioni tra i vari stati.

5.12.2 Documentazione dei costruttori e dei distruttori

5.12.2.1 StateMachine()

```
StateMachine::StateMachine (
            state start)
```

Costruttore della classe.

Costruttore della classe StateMachine

Parametri

start	Lo stato con cui partirà il gioco

5.12.3 Documentazione delle funzioni membro

5.12.3.1 flip()

```
void StateMachine::flip ()
```

Cambia stato di gioco.

Cambia lo stato attuale del gioco

5.12.3.2 game_loop()

```
void StateMachine::game_loop ()
```

Ciclo principale del gioco.

Fa partire il gioco intero attraverso un loop infinito

5.12.3.3 init_ncurses()

```
void StateMachine::init_ncurses ()
```

Inizializza le funzioni di ncurses.

Inizializza tutte le funzioni di ncurses necessarie

5.12.3.4 update()

```
void StateMachine::update (
            int input)
```

Aggiorna lo stato attuale.

Aggiorna lo stato attuale e controlla che non ci sia bisogno di cambiarlo

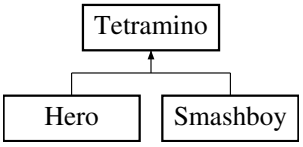
La documentazione per questa classe è stata generata a partire dai seguenti file:

- include/state_machine/state_machine.hpp
- include/state_machine/state_machine.cpp

5.13 Riferimenti per la classe Tetramino

```
#include <tetramini.hpp>
```

Diagramma delle classi per Tetramino



Membri pubblici

- Tetramino (World world, int w, int h)
  Costruttore della classe.
- void print (int shape)
  Stampa il tetramino nel terminale.
- virtual void print_frame ()=0
  Funzione astratta.
- void move (int dir)
  Muove in tetramino a destra o a sinistra.
- virtual void safe_move (int dir)=0
  Funzione astratta.
- int falling ()
  Simulazione della caduta del tetramino.
- void dies ()
  Morte del tetramino.
- virtual void rotate ()=0
  Funzione astratta.
- int check_collision ()
  Controlla le collisioni del tetramino.
- void getclout (int row, char *buffer)
  Fa sapere il contenuto di una riga del tetramino.

Attributi protetti

- int **SCRW**
- int **SCRH**
- int **XOFF**
- int **WIDTH**
- int **HEIGHT**
- int **STARTX** =20
- int **STARTY** =1
- int **x**
- int **y**
- WINDOW * **base**

5.13.1 Descrizione dettagliata

Classe astratta di un tetramino generico.

5.13.2 Documentazione dei costruttori e dei distruttori

5.13.2.1 Tetramino()

```
Tetramino::Tetramino (
            World world,
            int w,
            int h)
```

Costruttore della classe.

Costruttore della classe

Parametri

world	il mondo in cui esisterà il tetramino
w	la larghezza del tetramino
h	l'altezza del tetramino

5.13.3 Documentazione delle funzioni membro

5.13.3.1 check_collision()

```
int Tetramino::check_collision ()
```

Controlla le collisioni del tetramino.

Controlla che le vicinanze del tetramino siano libere

5.13.3.2 dies()

```
void Tetramino::dies ()
```

Morte del tetramino.

Quando cade il tetramino muore.

5.13.3.3 falling()

```
int Tetramino::falling ()
```

Simulazione della caduta del tetramino.

Simula la gravità nel mondo e fa cadere il tetramino verso terra

5.13.3.4 getclout()

```
void Tetramino::getclout (
            int row,
            char * buffer)
```

Fa sapere il contenuto di una riga del tetramino.

Prende il contenuto di una riga del tetramino

Parametri

in	*la*	riga in questione
out	∗*buffer*	la stringa di output

5.13.3.5 move()

```
void Tetramino::move (
            int dir)
```

Muove in tetramino a destra o a sinistra.

Muove il tetramino a destra o a sinistra nel terminale

Parametri

dir	positiva se a destra negativa altrimenti

5.13.3.6 print()

```
void Tetramino::print (
            int shape)
```

Stampa il tetramino nel terminale.

Stampa il tetramino nel terminale

Parametri

shape	la forma del tetramino

5.13.3.7 print_frame()

```
virtual void Tetramino::print_frame ()  [pure virtual]
```

Funzione astratta.

Implementato in Hero, e Smashboy.

5.13.3.8 rotate()

```
virtual void Tetramino::rotate ()  [pure virtual]
```

Funzione astratta.

Implementato in Hero, e Smashboy.

5.13.3.9 safe_move()

```
virtual void Tetramino::safe_move (
            int dir)  [pure virtual]
```

Funzione astratta.

Implementato in Hero, e Smashboy.

La documentazione per questa classe è stata generata a partire dai seguenti file:

- include/tetramini/tetramini.hpp
- include/tetramini/tetramini.cpp

5.14 Riferimenti per la classe World

`#include <world.hpp>`

Membri pubblici

- World (int w, int h, int xoff)

  Costruttore della classe.
- void getspecs (int ∗w, int ∗h, int ∗xoff)

  Fa sapere le specifiche del mondo di gioco.
- void pos_to_coords (int pos, int ∗x, int ∗y)

  Converte la posizione nella griglia in coordinate reali.
- void draw ()

  Stampa il mondo nel terminale.
- void scan ()

  Scannerizza tutto il mondo e aggiorna la griglia.
- int checkfullrow ()

  Controlla se ci sono righe piene.
- void update_screen ()

  Aggiorna la finestra.
- void update_points (int p)

  Aggiorna il punteggio a schermo.

5.14.1 Descrizione dettagliata

Il mondo di tetris che gestisce le interazioni tra tetramino e l'esterno

5.14.2 Documentazione dei costruttori e dei distruttori

5.14.2.1 World()

```
World::World (
            int w,
            int h,
            int xoff)
```

Costruttore della classe.

Costruttore della mondo di gioco

Parametri

w	la larghezza del mondo
h	l'altezza del mondo
xoff	quanto a destra è spostato il mondo

5.14.3 Documentazione delle funzioni membro

5.14.3.1 checkfullrow()

```
int World::checkfullrow ()
```

Controlla se ci sono righe piene.

Controllare se ci sono righe piene nel mondo

Restituisce

i punti guadagnati per aver fatto quel numero di righe piene

5.14.3.2 draw()

```
void World::draw ()
```

Stampa il mondo nel terminale.

Stampa tutto il mondo nel terminale

5.14.3.3 getspecs()

```
void World::getspecs (
            int * w,
            int * h,
            int * xoff)
```

Fa sapere le specifiche del mondo di gioco.

Per sapere le specifiche del mondo di gioco

Parametri

out	*w*	la larghezza del mondo
out	*h*	l'altezza del mondo
out	*xoff*	quanto è spostato a destra il mondo

5.14.3.4 pos_to_coords()

```
void World::pos_to_coords (
            int pos,
            int * x,
            int * y)
```

Converte la posizione nella griglia in coordinate reali.

Converte la posizione all'interno della griglia nelle coordinate effitive della finestra

Parametri

in	*la*	posizione nella griglia
out	*ascissa*	nella finestra
out	*ordinata*	nella finestra

5.14.3.5 scan()

```
void World::scan ()
```

Scannerizza tutto il mondo e aggiorna la griglia.

Scannerizza tutto il mondo e aggiorna la griglia

5.14.3.6 update_points()

```
void World::update_points (
            int p)
```

Aggiorna il punteggio a schermo.

Aggiorna il punteggio a schermo

Parametri

p	i nuovi punti

5.14.3.7 update_screen()

```
void World::update_screen ()
```

Aggiorna la finestra.

Aggiorna i bordi della finestra

La documentazione per questa classe è stata generata a partire dai seguenti file:

- include/states/world.hpp
- include/states/world.cpp

Capitolo 6

Documentazione dei file

6.1 state.hpp

```
00001 #ifndef STATE_HPP
00002 #define STATE_HPP
00003
00004 #define STATES_LEN 3
00005
00010 typedef enum states_list {
00011     MENU,
00012     PARTITA,
00013     CLASSIFICA
00014 } state;
00015
00020 class State {
00021     protected:
00022         int done = 0;
00023         int quit = 0;
00024         state next;
00025         state previous;
00026
00027     public:
00028
00030         State() {}
00031
00033         void setDone(int d);
00034
00036         int isDone();
00037
00039         void setNext(state n);
00040
00042         state getNext();
00043
00045         void setPrev(state prev);
00046
00048         void Quit();
00049
00051         int getQuit();
00052
00054         virtual int update(int input)=0;
00055 };
00056
00057 #endif
```

6.2 state_machine.hpp

```
00001 #ifndef STATE_MACHINE_HPP
00002 #define STATE_MACHINE_HPP
00003
00004 #include <ctime>
00005 #include <ncurses.h>
00006
00007 #include "state.hpp"
00008
00009 #include "../states/menu.hpp"
00010 #include "../states/gameplay.hpp"
```

```
00011 #include "../states/classifica.hpp"
00012
00020 class StateMachine {
00021     protected:
00022         int done = 0;
00023
00024         state current_state;
00025         State *states[STATES_LEN];
00026         State *current;
00027
00028     public:
00030         StateMachine(state start);
00031
00033         void init_ncurses();
00034
00036         void flip();
00037
00039         void update(int input);
00040
00042         void game_loop();
00043 };
00044
00045
00046 #endif
```

6.3 classifica.hpp

```
00001 #ifndef CLASSIFICA_HPP
00002 #define CLASSIFICA_HPP
00003
00004 #include <string.h>
00005 #include <fstream>
00006 #include <ncurses.h>
00007 #include "../state_machine/state.hpp"
00008 #include "screen.hpp"
00009 using namespace std;
00010
00011 typedef struct pnode {
00012     pnode *next;
00013     char all[100];
00014     char name[50];
00015     int points;
00016 } player;
00017
00021 class Classifica : public State {
00022     private:
00023         const char *filename = "classifica.txt";
00024         const char splitter = '@';
00025         ofstream ofile;
00026         ifstream ifile;
00027
00028         player *head = new player;
00029
00030         char *all_players;
00031
00032         List chart;
00033
00034         int ROW_SPACING = 2;
00035         int TEXT_XOFF = 3;
00036         int TEXT_YOFF = 1;
00037
00038         int total_len=0;
00039         int length=0;
00040
00041     public:
00042
00044         Classifica();
00045
00047         void display();
00048
00050         int update(int input);
00051
00053         player gameOverScreen(int points);
00054
00056         void get_new_name(char name[]);
00057
00059         void addPlayer();
00060
00062         void insertPlayer(char name[], int points);
00063 };
00064
00065 #endif
```

6.4 gameplay.hpp

```
00001 #ifndef GAMEPLAY_HPP
00002 #define GAMEPLAY_HPP
00003
00004 #include <ctime>
00005 #include <fstream>
00006 #include <ncurses.h>
00007 #include "world.hpp"
00008 #include "../state_machine/state.hpp"
00009 #include "../tetramini/tetramini.hpp"
00010 #include "../tetramini/hero.hpp"
00011 #include "../tetramini/smashboy.hpp"
00012 using namespace std;
00013
00017 class Partita : public State {
00018     private:
00019         World *world;
00020
00021         double timer;
00022         clock_t tm;
00023         double velocity;
00024
00025         int tick;
00026
00027         /*
00028          * si utilizza un puntatore alla classe base
00029          * che punterà casualmente, a turno,
00030          * una variabile della classe derivata
00031          */
00032         Tetramino *t;
00033         Hero *h;
00034         Smashboy *s;
00035
00036         int input;
00037         int is_moving=0;
00038         int points=0;
00039         int game_over=0;
00040         int falls;
00041
00042     public:
00044         Partita();
00045
00047         int update(int input);
00048
00050         int gameplay(int input);
00051
00053         void gameOver();
00054 };
00055
00056 #endif
```

6.5 menu.hpp

```
00001 #ifndef MENU_HPP
00002 #define MENU_HPP
00003
00004 #include <ncurses.h>
00005 #include "../state_machine/state.hpp"
00006 #include "screen.hpp"
00007
00012 class Menu : public State {
00013     private:
00014         int ROW_SPACING = 2;
00015
00016         List options;
00017
00018         int TEXT_XOFF = 3;
00019         int TEXT_YOFF = 1;
00020
00021         const char CRS_CH = '>';
00022         int CRS_YOFF;
00023         int CRS_X;
00024         int CRS_Y;
00025         int cursor = 0;
00026
00027     public:
00029         Menu();
00030
00032         void draw();
00033
00035         int update(int input);
```

```
00036 };
00037
00038 #endif
```

6.6 screen.hpp

```
00001 #ifndef SCREEN_HPP
00002 #define SCREEN_HPP
00003
00004 #include <string.h>
00005 #include <ncurses.h>
00006
00011 typedef struct f {
00012     char *str[50];
00013     int max_len;
00014 } field;
00015
00019 class Screen {
00020     protected:
00021         int WIDTH, HEIGHT;
00022         int STARTY, STARTX;
00023         char BORDERS[8];
00024
00025     public:
00026
00027         WINDOW *win;
00028
00030         Screen(){};
00031
00033         void init(int w, int h, int stx, int sty, char borders[]);
00034
00036         void setInCenter();
00037
00039         void show();
00040
00042         void destroy();
00043 };
00044
00048 class List : public Screen {
00049     private:
00050         int LENGTH;
00051         int ROW_SPACING;
00052         int TEXT_XOFF;
00053         int TEXT_YOFF;
00054
00055         int TOTAL_CHARS_LEN;
00056
00057         char *list = NULL;
00058         char splitter;
00059         const char cols_d = '@';
00060         const char rows_d = '\n';
00061
00062         int NFIELDS=0;
00063
00064         field *f;
00065
00066     public:
00067
00069         List(){};
00070
00072         void init(char text[], char delimiter, int r_s, int txt_xoff, int txt_yoff, char borders[]);
00073
00075         void compute_sizes(int *width, int *height);
00076
00078         void update_list(char new_text[]);
00079
00081         void show_list();
00082
00084         int get_len();
00085 };
00086
00087 #endif
```

6.7 world.hpp

```
00001 #ifndef GRID_HPP
00002 #define GRID_HPP
00003
```

```
00004 #include <ncurses.h>
00005
00010 typedef struct node {
00011     node *next;
00012     int val;
00013     int row_blocks;
00014 } block;
00015
00020 class World {
00021     private:
00022         //dimensioni dello schermo di gioco
00023         int SCRW, SCRH;
00024
00025         //differenza di posizione rispetto all'origine
00026         int XOFF;
00027
00028         WINDOW *screen;
00029
00030
00031         //dimesioni dello schermo dei punti
00032         int PNT_SCRW = 25;
00033         int PNT_SCRH = 3;
00034
00035         //posizione dello schermo dei punti
00036         int PNT_SCRX = 50;
00037         int PNT_SCRY = 2;
00038
00039         WINDOW *points_scr;
00040
00041
00042         /*
00043          * differenze di posizione dei blocchi
00044          * rispetto allo schermo di gioco
00045          */
00046         int GRID_XOFF=1;
00047         int GRID_YOFF=1;
00048
00049         //dimensioni della griglia
00050         int GRIDW;
00051         int GRIDH;
00052         int LEN;
00053
00054         //la griglia è una lista concatenata
00055         block *grid;
00056
00057     public:
00059         World(int w, int h, int xoff);
00060
00062         void getspecs(int *w,  int *h, int *xoff);
00063
00065         void pos_to_coords(int pos, int *x, int *y);
00066
00068         void draw();
00069
00071         void scan();
00072
00074         int checkfullrow();
00075
00077         void update_screen();
00078
00080         void update_points(int p);
00081
00082 };
00083
00084 #endif
```

6.8 hero.hpp

```
00001 #ifndef HERO_HPP
00002 #define HERO_HPP
00003
00004 #include <iostream>
00005 #include "tetramini.hpp"
00006 #include "../states/world.hpp"
00007
00008 #define FRLEN 2
00009
00013 class Hero: public Tetramino {
00014     protected:
00015         long int frames[FRLEN] = {
00016             61440, 8738
00017             //4369, 8738, 17476, 34952,
00018             //15, 240, 3840, 61440
```

```
00019         };
00020         int current=0;
00021     public:
00023         Hero(World world);
00024
00026         void print_frame();
00027
00029         void safe_move(int dir);
00030
00032         void rotate();
00033
00035         int side_collisions();
00036 };
00037
00038 #endif
```

6.9 smashboy.hpp

```
00001 #ifndef SMASHBOY_HPP
00002 #define SMASHBOY_HPP
00003
00004 #include "../states/world.hpp"
00005
00009 class Smashboy: public Tetramino {
00010     protected:
00011         int frame = 15;
00012         //char frame[5] = "1111";
00013
00014     public:
00016         Smashboy(World world);
00017
00019         void print_frame();
00020
00022         void safe_move(int dir);
00023
00025         void rotate() {}
00026 };
00027
00028 #endif
```

6.10 tetramini.hpp

```
00001 #ifndef TETRAMINI_HPP
00002 #define TETRAMINI_HPP
00003
00004 #include <ncurses.h>
00005 #include "../states/world.hpp"
00006
00010 class Tetramino {
00011     protected:
00012         int SCRW;
00013         int SCRH;
00014         int XOFF;
00015
00016         int WIDTH;
00017         int HEIGHT;
00018
00019         int STARTX=20;
00020         int STARTY=1;
00021
00022         int x;
00023         int y;
00024
00025         WINDOW *base;
00026     public:
00028         Tetramino(World world, int w, int h);
00029
00031         void print(int shape);
00032
00034         virtual void print_frame()=0;
00035
00037         void move(int dir);
00038
00040         virtual void safe_move(int dir)=0;
00041
00043         int falling();
00044
00046         void dies();
```

```
00047
00049         virtual void rotate()=0;
00050
00052         int check_collision();
00053
00055         void getclout(int row, char *buffer);
00056 };
00057
00058 #endif
```

Indice analitico