Tetris

Alessandro Roncaglia, **Andrea Dal Cin**
Domentico Livera, **Elena Bartolotta**

June 2024

1 Introduzione

1.1 Obiettivo

L'obiettivo di questo progetto è realizzare il noto gioco "Tetris" utilizzando la libreria grafica "ncurses.h" creando così un gioco eseguibile da riga di comando e giocabile sul terminale. Il giocatore deve avere la possibilità scegliere tra:

- "avviare una nuova partita"
- "visualizzare la classifica"
- "uscire definitivamente dal gioco"

2 Struttura

2.1 Main()

1. La prima cosa che viene inizializzata nel main è ovviamente lo schermo, su cui chiamiamo il metodo noecho() in modo che l'input dell'utente non venga mostrato. Dopo aver inizializzato i colori, basandosi sulla dimensione dello standard screen, creiamo e posizioniamo la finestra principale **mainWin**. Dichiariamo gli oggetti delle classi Option, Menu, Hud, TetrisBoard, Tetramino, Classifica e Game.

2. Il while si occupa del flusso dell'esecuzione del programma. Invoca il metodo draw del menu e aspetta che l'utente scelga una delle tre opzioni. La scelta viene gestita tramite uno switch e nei diversi casi vengono invocati i metodi che si occupano dell'opzione scelta.

3. Caso **NEW GAME**:
   Vengono stampate le istruzioni per il gioco e la finestra contenente il tetramino successivo. Viene invocato il metodo loop di game che dà inizio alla caduta dei tetramini, e con questa del gioco stesso. Una volta che si esce dal loop di gioco il punteggio viene memorizzato nella variabile score

che viene passata al metodo della classifica in modo da aggiornarla. Il reset del campo da gioco e la distruzione della HUD avvengono all'interno del metodo loop della classe GAME.

4. Caso **LEADERBOARD**:
   Nel caso leaderboard viene semplicemente stampata la classifica tramite il metodo Mostra().

2.2 Classi implementate

- **Utils**:
  è un file hpp contenente varie costanti del gioco: sono presenti i 7 tipi di tetramini rappresentati come matrici intere 4x4, le variabili intere che indicano le dimensioni della board di gioco, tre costanti stringa che indicano le voci del menu, le costanti intere che indicano i colori, e infine il moltiplicatore che viene applicato quando viene calcolato il punteggio se più linee vengono completate contemporaneamente.

- **Drawable**:
  Classe madre di tutti gli elementi che vengono proiettati durante l'esecuzione. I suoi attributi sono le coordinate **x**, **y** dell'oggetto e la finestra su cui viene disegnato. Escludendo costruttori, getter e setter il metodo principale è **draw()**, che viene implementato diversamente da ogni classe figlia.

- **Tetramino**:
  Estende la classe **Drawable** e implementa i vari tetramini. Ha due attributi propri: un intero **color** che rappresenta il colore e un array bidimensionale 4x4 **shape** che descrive la forma del tetramino. I suoi metodi sono: **moveLeft**, **moveRight**, **moveDown**, **moveUp** che incrementano o diminuiscono i valori **x**,**y** del tetramino; **rotateLeft** e **rotateRight** che ruotano la matrice shape di 90°, il metodo **spawn**, che prende casualmente uno dei possibili 7 tipi di tetramino presenti nel file **utilis.hpp** e lo assegna all'attributo **shape** dell'oggetto assieme al colore associato al specifico tetramino.

- **TetrisBoard**:
  Estende anch'essa la classe **Drawable** e implementa la griglia di gioco dove cadono i tetramini, ha quindi come attributo l'array intero bidimensionale board. Implementa le funzioni per il controllo delle collisioni: **canPlaceTetramino** verifica che ci sia spazio nella board per disegnare il tetramino; **isHittingFloor**, **isHittingRightWall**, **isHittingLeftWall** restituiscono il valore booleano true se il tetramino è a contatto con uno dei margini della board. Il metodo **pinTetramino** viene chiamato quando un tetramino finisce la sua discesa e viene quindi "fissato" sulla board. Sono presenti poi i metodi booleani **isLineFull** e **isLineEmpty** necessari per l'implementazione del metodo **clearLines** che rimuove tutte le linee complete dalla board.

- **Game**:
  Racchiude il cuore del gioco. I campi comprendono la **finestra principale,** il campo da gioco "**board**" appartenente alla classe **TetrisBoard**, un oggetto della classe **Tetramino** e **l'hud**. Oltre al costruttore presenta solo un metodo che è il metodo **loop()** che una volta invocato fa sì che inizi il loop della caduta dei tetramini. All'interno di questo loop è presente uno switch che processa gli input dell'utente. Tale loop viene interrotto in due casi: nel caso in cui si prema il tasto **q**, o nel caso in cui l'ultimo tetramino comparso esca dal confine superiore della board. In entrambi i casi il punteggio conseguito viene restituito dalla funzione ed è successivamente memorizzato nella **classifica**.

- **Hud**:
  La classe Hud contiene i campi che rappresentano il **punteggio**, un **moltiplicatore** per il punteggio, il numero di **linee** pulite in una singola mossa, la **finestra** su cui viene disegnato l'hud, la **finestra** su cui viene disegnato il prossimo **Tetramino** e L'effettivo **Tetramino** successivo. La funzione principale è **printHUD()** la quale stampa: Le istruzioni che riguardano il funzionamento di ogni tasto premuto e il prossimo Tetramino, che apparirà appena quello attuale entrerà in collisione con un altro o con il pavimento, grazie alle due funzioni **nextPiece printNextShape()** le quali impostano quale sarà il prossimo **pezzo** e lo stampano a schermo sfruttando la funzione **draw()** ereditata dalla classe **Drawable,** Il punteggio viene calcolato attraverso la funzione **computeScore()**. Infine la funzione **destroyHUD()** pulisce e refresha tutte le finestre create dalla classe deallocandone la memoria.

- **Option**:
  La classe Option è la classe di ogni singolo oggetto opzione che ha come campi: il nome e le due coordinate dove verrà stampata l'opzione stessa. I metodi, a parte il costruttore che inizializza il nome dell'opzione, si occupano di impostare e restituire le coordinate dell'opzione.

- **Menu**:
  La classe **Menu** contiene un campo di tipo **WINDOW** dove viene passata la finestra principale in cui verrà scritto il menù, un campo di tipo **Option** dove vengono passate le opzioni e il **numero** delle opzioni. La scrittura del menu sulla finestra avviene tramite il metodo **draw.**
  Il costruttore della classe, oltre ad inizializzare i campi principali, si occupa di assegnare le coordinate sullo schermo dove verranno scritte le opzioni.
  Il metodo **draw** fa sì, tramite un ciclo while, che l'utente possa muoversi tra le opzioni usando le frecce. Una volta che l'utente avrà premuto enter, il metodo **draw()** restituirà il numero dell'opzione evidenziata nel momento in cui è stato premuto **enter** (o invio).

- **Classifica**:
  La classe **Classifica**, come la classe **Menu**, ha tra i suoi campi una finestra che ha funzione analoga a quella del menu. In più ha un puntatore

ad una lista di punteggi. Punteggio è un struttura dichiarata in questa stessa classe, e ha un parametro **p** di tipo stringa, che contiene il punteggio e un puntatore a punteggio. I punteggi vengono salvati su un file di testo. Il metodo **Mostra()** apre questo file per leggerne il contenuto e stamparlo sulla finestra. Per uscire l'utente deve premere **enter** (o invio). Il salvataggio dei punteggi in ordine decrescente è affidato al metodo **Aggiorna()** che prende in input il punteggio appena conseguito, e lo inserisce nella lista di punteggi salvati precedentemente. A questo punto apre il file di testo e lo riscrive da capo. Aggiorna() sfrutta due funzioni ausiliarie: **FromFileToList()** e **sortInsert().**
FromFileToList(): prende ogni linea del file di testo e costruisce una lista ordinata di elementi di tipo Punteggio.
sortInsert(): inserisce il punteggio conseguito all'interno di una lista ordinata, mantenendo l'ordine.

3 Scelte implementative

- **Scelta opzioni**: A seguito della scelta dell'utente la finestra viene pulita con il metodo **wclear()** e riscritta attraverso il metodo della classe che si occupa dell'opzione scelta: in generale il passaggio da una finestra all'altra è dato da operazioni di pulitura e riscrittura di un'unica finestra principale (**mainWin**) che viene creata nel **main**.
- **Rotazione e movimento**: La rotazione e il movimento dei tetramini avvengono all'interno della classe game, nello switch che gestisce gli input da tastiera dell'utente. Quando il giocatore preme un tasto per muovere il tetramino o ruotarlo viene chiamato il rispettivo metodo sull'oggetto tetramino, dopodichè viene effettuate un controllo delle collisioni con i metodi della board isHittingFloor/Wall e canPlaceTetramino, e se vengono riscontrate delle collisioni si effettua il movimento opposto a quello effettuato dall'utente per annullare la rotazione o il movimento. e.g:

```
case 's':
    this->tetramino.moveDown();
    if(!this->board.canPlaceTetramino(&this->tetramino)) {
        this->tetramino.moveUp();
    }
    break;
```

- **Calcolo del punteggio**: Il calcolo del punteggio viene effettuato tramite la funzione computeScore() il quale aumenta il campo punteggio aggiungendo le linee cancellate con una sola mossa, ottenute dal metodo setLines(int lines) dove "lines" è il valore restituito dal metodo clearLines() della classe TetrisBoard, moltiplicate per il campo multiplier.