**Struttura del Progetto**

Il progetto è stato strutturato utilizzando un'architettura a classi, suddividendo le responsabilità tra diverse classi per facilitare la gestione del codice e la sua estensibilità. Le principali classi implementate sono:

1. **Tetramino**: Classe base per i vari pezzi del Tetris.
2. **Line, Square, IShape, OShape, TShape, LShape, JShape, SShape**: Classi derivate da Tetramino che rappresentano i diversi tipi di pezzi.
3. **Board**: Classe che gestisce la griglia di gioco.
4. **Game**: Classe principale che gestisce il flusso del gioco.
5. **GameInfo**: Classe che gestisce lo stato del gioco (livello, score ecc..)
6. **Leaderboard**: Classe che gestisce la scrittura e lettura del file dedicato al salvataggio dei punteggi.

**Scelte di Implementazione**

**Pagine principali**

Per quanto riguarda la pagina iniziale, quella di game over e quella di lettura dei punteggi abbiamo deciso di creare delle window dedicate (ognuna con la propria classe dedicata), implementando un menù navigabile da tastiera che evidenzia l’opzione selezionata in pieno stile arcade game. Per farlo, abbiamo utilizzato le funzioni **wattron** e **wattroff** di ncurses.

**Classe Tetramino**

La classe Tetramino è la classe base per tutti i pezzi del Tetris. Contiene la dichiarazione delle informazioni comuni a tutti i pezzi, come la loro posizione all’interno della griglia e l’array rappresentante le possibili rotazioni del pezzo. La decisione di centralizzare queste informazioni in una classe base permette di ridurre la duplicazione di codice e semplificare l'implementazione delle singole forme.

**Implementazione delle Forme Specifiche**

Ogni forma specifica (IShape, OShape, ecc.) eredita dalla classe Tetramino e inizializza la propria matrice di rappresentazione e le possibili rotazioni. Questo approccio facilita l'aggiunta di nuove forme ogni qualvolta si voglia. I tetramini sulla griglia di gioco vengono renderizzati mediante l’utilizzo del carattere “X”.

**Gestione della Griglia di Gioco**

La classe Board gestisce la griglia di gioco, inclusa la verifica della presenza di linee complete e la loro eliminazione. Questa separazione delle responsabilità permette alla classe Game di concentrarsi sulla logica di gioco senza doversi preoccupare della gestione dettagliata della griglia.

La logica di gestione della griglia di gioco è stata demandata ad un **array bidimensionale**, contenente righe e colonne della griglia. In questo modo tramite questo semplice array è possibile in ogni momento sapere se una determinata cella è occupata da un pezzo o meno, e quindi gestire eventuali collisioni e completamento di linee.

La griglia di gioco viene renderizzata mediante l’utilizzo del carattere “.” mentre per la definizione dei bordi è stato sfruttato il metodo “**box**” di ncurses. Questa scelta è stata fatta per consentire una comprensione più chiara dello spazio di gioco da parte del giocatore, in particolare per evidenziare più facilmente il numero di spazi disponibili per incastrare i vari tetramini.

**Gestione del Flusso di Gioco**

La classe Game gestisce il flusso principale del gioco, incluse la generazione casuale dei pezzi, il controllo degli input dell'utente e l'aggiornamento dello stato del gioco.  
All’interno del loop principale di gioco avvengono principalmente 3 cose:

1. Si controlla un eventuale tasto premuto dal giocatore per muovere il tetramino, e in caso venga rilevato si procede con la sua gestione (controllo di possibilità di posizionamento del pezzo, rilevamento collisioni ecc…)
2. Ogni X millisecondi (in base al livello di gioco) il tetramino viene fatto scendere in automatico
3. Ogni secondo si aggiorna il tempo trascorso dall’inizio della partita all’interno della schermata contenente le info di gioco.

Abbiamo pensato che potesse essere divertente aumentare la **difficoltà** del gioco man mano che si procede con i livelli; per farlo abbiamo introdotto una variabile che tiene conto del numero di righe completate e abbiamo aumentato sempre di più la velocità di discesa del tetramino mediante la formula “1000”/(level\*1.5), dove 1000 rappresenta il tempo iniziale di discesa del tetramino a livello 1, ovvero ogni secondo.

Naturalmente non potevamo far si che una riga completata e un tetris fornissero lo stesso punteggio al giocatore quindi abbiamo pensato che fosse importante fornire più punti in base alle righe che vengono completate **contemporaneamente**, per farlo abbiamo usato la formula 10\*(righecompletate^2) in modo tale da fornire molti più punti per tetris che per una riga singola.

A fine partita viene poi salvato il punteggio del giocatore nel file contenente tutti i punteggi e mostrata una window di Game Over, dove il giocatore può decidere se tornare al menù principale o chiudere il gioco.

**Gestione delle Info di gioco**

La classe GameInfo gestisce le info relative al gioco attualmente in corso quali livello, punteggio, linee completate e tempo dall’inizio del gioco.

Le classi Game e GameInfo rappresentano due window di ncurses diverse, renderizzate graficamente l’una accanto all’altra. Si è deciso di fare questa scelta principalmente per differenziare le modalità di rendering delle finestre. La window di GameInfo, difatti, viene principalmente aggiornata ogni secondo (per tenere traccia del tempo trascorso) e ogni qualvolta venga completata una linea di gioco, mentre la window relativa al Game viene aggiornata ogni X millisecondi (in base al livello in cui ci si trova) per garantire la discesa del pezzo in maniera automatica e costante.

**Implementazione del Movimento Automatico e della Generazione dei Pezzi**

Uno degli aspetti più critici del gioco Tetris è il movimento automatico dei pezzi verso il basso. Questo è stato implementato utilizzando un ciclo principale che verifica periodicamente se il pezzo corrente può spostarsi verso il basso. Se non può, viene fissato nella sua posizione attuale (all’interno dell’array bidimensionale citato poco fa) e viene generato un nuovo pezzo casuale.

**Gestione della Leaderboard**

Tutti i punteggi registrati dal giocatore a fine game vengono salvati all’interno di un file “leaderboard.txt”. Per la lettura e la scrittura dei file è stata utilizzata la libreria fstream, che permette di leggere e scrivere su file di testo.  
Il nuovo punteggio a fine partita viene inserito in fondo al file.

In fase di lettura poi, tutti i punteggi vengono letti e ordinati in maniera decrescente per mostrarli ordinati all’interno della schermata.

Nel caso in cui il file non venga trovato (caso in cui è la prima volta che si avvia il gioco) a schermo verrà mostrata la scritta “no games finished”. E’ stato anche scelto di non mostrare i punteggi uguali a 0, che durante la lettura del file vengono ignorati, così come le eventuali righe vuote presenti nel file.

In fase di renderizzazione della leaderboard è stato scelto di implementare un sistema di **paginazione**, mostrando un massimo di 10 punteggi per pagina (questo per consentire una corretta visualizzazione anche su monitor di piccole dimensioni), gestendo il cambio di pagina tramite pulsanti “next page” e “previous page” (sempre mediante l’utilizzo delle funzioni **wattron** e **wattroff** di ncurses).

Viene mostrato anche il numero di punteggi totali presenti e un’indicazione riguardante il numero di pagina in cui ci si trova e il numero di pagine totali.