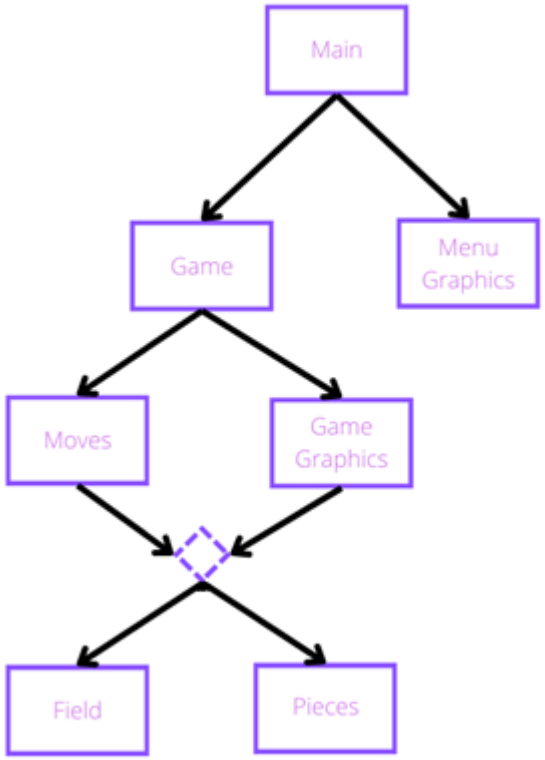
Progetto X-Tetris

Contenuti della relazione

Struttura del progetto

Il progetto è diviso in diverse unità. Le unità più semplici svolgono compiti specifici e vengono combinate da unità più complesse per svolgere compiti avanzati. La struttura si articola come segue:



- Il programma principale usa le librerie MenuGraphics e Game, che gestiscono rispettivamente la grafica per il menu iniziale, da cui si seleziona la modalità, e la partita in sé, diversa a seconda della modalità scelta.
- La libreria Game combina la grafica della partita (quindi punteggi, campi, tetramini…) con la logica della libreria Moves, combinando e alternando le rispettive funzioni. Questa libreria per l'appunto lavora con campi e tetramini per movimenti come la rotazione e l'inserimento di un tetramino nel campo.
- Le librerie Moves e GameGraphics sfruttano entrambe le proprietà e le funzioni delle librerie Field e Pieces, che rappresentano rispettivamente il campo e i tetramini, con funzioni come l'inizializzazione, la deallocazione e l'ottenimento di altre proprietà, come ad esempio la quantità disponibile del tetramino o la dimensione del campo.
- Altre librerie di supporto sono state usate ma non sono determinanti per quanto riguarda la struttura generale del programma.

Implementazione

Il campo

Il campo è stato modellato con una matrice di interi (15 + 4)x(10). Al numero 0 corrisponde una cella vuota nel campo, mentre valori maggiori di 0 sono considerati come celle occupate da tetramini.

Le righe "valide" sono 15 e sono quelle righe che sono stampate, visibili all'utente e giocabili. Le restanti 4 righe sono invisibili e si trovano sopra il limite di altezza del campo. Se dopo una mossa un tetramino si trova in una di queste righe, la partita sarà considerata persa. È stato scelto il numero 4 perché è l'altezza massima di un tetramino.

Particolari funzioni della libreria grafica si occuperanno di tradurre il campo da array di interi a stringhe colorate stampabili sul terminale.

I tetramini

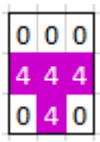
I tetramini sono 7. Ciascuno di essi è codificato con una struct che ha diversi campi. I più notevoli sono il campo `value` e il campo `shape`. Il campo `value` è il valore che identifica un tetramino all'interno del campo, ovvero all'interno dell'array. Ad esempio una cella con il valore 1 è identificativa di un certo tetramino, mentre una cella con valore 2 sarà rappresentativa di un altro tetramino (ricordando che una cella con valore 0 rappresenta una cella vuota, libera). Il campo `shape` invece rappresenta la forma corrente di un tetramino, ovvero la forma base con le rotazioni applicate. La forma è modellata con una matrice quadrata, di dimensioni diverse in base al tetramino. La dimensione è giusta da poter contenere ogni rotazione possibile del tetramino. La matrice è in realtà un array flattened perché allocato in memoria dinamica.

L'inserimento

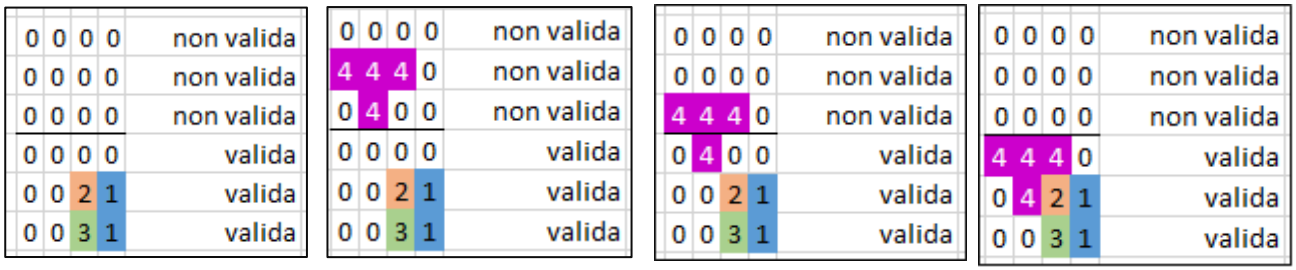
Implementato nella libreria Moves.h, questa funzione combina campo e tetramini. Scelta la rotazione e la colonna, il tetramino viene inserito nel campo. Immaginando la forma di un tetramino come una matrice quadrata, posizioniamo il tetramino nel campo di gioco, a partire dalla primissima riga (che è una delle righe non valide). Per quanto riguarda il posizionamento orizzontale, la prima colonna del tetramino verrà inserita nella colonna scelta per l'inserimento. La forma viene sempre aggiustata dopo ogni rotazione in modo che la prima colonna del tetramino non sia vuota.

Dopodiché si procede come per il tetris originale. Il tetramino viene fatto "cadere" riga per riga, finché non incontra dei pezzi di altri tetramini. A quel punto la discesa si ferma e l'inserimento viene completato.

Di seguito un esempio con un campo ridotto. Nota: i colori sono gestiti dalla libreria grafica, qui sono solo usati per avere una maggiore visibilità. La funzione che si occupa dell'inserimento non fa nulla di grafico.



Supponiamo di avere questo tetramino. La dimensione della matrice permette di contenere tutte le rotazioni



Il campo ha già dei pezzi al suo interno. Il tetramino viene inserito nella prima riga e nella prima colonna. Dopo l'inserimento (seconda immagine) e il secondo passaggio (terza immagine) non sono state rilevate collisioni, quindi si controlla la riga successiva. Vediamo quindi che alla terza colonna abbiamo una collisione (due celle non vuote sono una sopra l'altra). Possiamo quindi procedere con l'inserimento del tetramino nel campo di gioco.

Grafica e Input

La libreria ncurses.h offre funzioni di grafica e utilità avanzate come menu e finestre su cui disegnare e un miglior input da tastiera. Queste funzionalità si possono però comunque implementare, anche se in maniera più complicata, senza installare librerie aggiuntive.

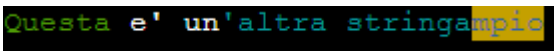
Colori con ANSI escape sequences

Colorare un terminale non è una prerogativa solo di alcune librerie. È possibile impostare dei colori al testo stampato a video tramite particolari sequenze di escape standard. Sono semplicemente sequenze di caratteri particolari che iniziano con il cosiddetto carattere di escape e una parentesi quadra. Il carattere di escape è indicato con "`\x1b`" che sarebbe l'esadecimale per 27 (tabella ASCII). Quello che viene dopo la parentesi quadra viene interpretato dal terminale in modi diversi, trattando i numeri come parametri e le lettere come il comando che il terminale deve eseguire. Ad esempio:

`\x1b[32m`	imposta il colore dei caratteri a verde
`\x1b[34;43m`	imposta il colore dei caratteri a verde e lo sfondo a giallo
`\x1b[0m`	reimposta il colore di default (lo 0 si può omettere)
`\x1b[18D`	sposta il cursore indietro di (18) posizioni

La particolarità delle sequenze di escape è che sono incorporate nel testo, solo che quando il terminale le analizza, al posto di stamparle esegue un comando. Finché i colori non vengono cambiati, ogni carattere sarà stampato colorato, anche per altri programmi. Qui ad esempio viene riportata una semplice `printf()`:

```
printf( "\x1b[32mQuesta \x1b[0me' una frase \x1b[34;43mdi "
        "esempio\x1b[0m\x1b[18D\x1b[36m'altra stringa\x1b[m");
```

La funzione avrà questo output → 

La libreria termios.h

Il sistema di input standard prevede funzioni come `scanf()` e `getchar()`. Quella che ci interessa è la seconda, in grado di ricevere in input singoli caratteri inseriti da tastiera. La funzione `getchar()` opera tramite un buffer. Se il buffer è vuoto, la funzione attende l'input dell'utente. Un tasto della tastiera può inserire uno o più caratteri nel buffer (esempio: le frecce direzionali sono identificate da una sequenza di escape di 3 caratteri). L'input termina dopo aver premuto il tasto invio (line feed), anch'esso inserito nel buffer. Se ci sono già caratteri nel buffer, `getchar()` ritorna il primo carattere del buffer senza chiedere input all'utente.
Questa funzione è ottima per l'implementazione dell'input da tastiera, ma abbiamo due problematiche principali

- Ogni tasto inserito viene anche stampato a schermo (echo)
- Il buffer è di dimensione variabile e ogni input termina sempre con il tasto invio

Il fatto che la funzione `getchar()` chieda un input solo se il suo buffer è vuoto, è un problema perché potrebbe considerare input vecchi nel momento sbagliato, e per confermare un input bisogna sempre premere il tasto invio. Per risolvere, l'input deve terminare ogni volta che si preme qualsiasi tasto.

La soluzione non è immediata e richiede l'uso della libreria termios.h, che permette di modificare dei flag che permettono al terminale di operare in maniera diversa. Bisogna disabilitare i flag ECHO e ICANON (che controlla che ogni input termini con il tasto invio). Il codice per farlo è il seguente:

```
struct termios term;                        /* variabile in cui salvare i flag   */
tcgetattr( STDIN_FILENO, &term);            /* ottiene i flag del terminale      */
term.c_lflag &= ~(ICANON | ECHO);           /* modifica i flag                   */
tcsetattr( STDIN_FILENO, TCSANOW, &term);   /* applica subito le modifiche       */
```

Materiale utile

[Tetris (Videogioco)](#)
[Modalità canonica termios](#)
[Come funziona getchar()](#)
[ANSI escape codes](#)