Relazione progetto XTetris

Daniel Andrei Bercu
Matricola 891470

1 Preambolo - Ai Professori e ai Tutor

Partendo da esperienza pari a zero nel settore della programmazione, questo è stato il mio primo progetto, sviluppato completamente dal nulla e portato avanti cercando di attenermi a soluzioni più "standard" possibile: nel caso in cui non l'abbiate ancora corretto al momento della lettura di questa relazione, Vi chiedo di tenere sempre a mente che si tratta di un progetto semplice, sicuramente pieno di bug di cui non sono nemmeno a conoscenza e/o che non ritengo di essere in grado di risolvere e, come ho detto nella prima riga, che si tratta del mio primo progetto.
È per queste stesse ragioni che voglio anche ringraziare i Professori Lucchese e Spanò per avermi lanciato questa sfida intrigante, che mi ha aperto gli occhi sui meccanismi e sul funzionamento del linguaggio C.
Vi auguro una buona lettura!

2 Ambiente di sviluppo, tempistiche e impostazione progetto

Il progetto è stato sviluppato complessivamente tra metà gennaio e fine aprile 2022, in maniera completamente individuale dal sottoscritto. Per i riferimenti e le influenze del progetto andare all'ultimo paragrafo della presente relazione.
Ho adottato *Sublime Text* come editor di testo e *gcc* come compilatore utilizzato via terminale in ambiente Manjaro Linux.
Il file system del progetto prevede che i file sorgenti siano locati nella directory *src/*, i file di intestazione in *include/* e la corrente relazione e la consegna del progetto in *report/*.
Con il comando `make` verranno generate tre directory: *build/*, contenente il file eseguibile, *obj/*, contenente i file oggetto, e *docs/*, contenente la documentazione generata con il software *Doxygen*.
Il codice, i commenti e la documentazione sono stati scritti in inglese perché, secondo me, aspetti più "generali" del progetto, mentre invece interfaccia utente del gioco e relazione sono stati scritti in italiano perché, a mio parere, più indirizzati a Professori e Tutor.

3 Problematiche riscontrate

Di seguito la lista di problemi che mi hanno fatto ragionare maggiormente sull'implementazione del progetto e le rispettive soluzioni.

3.1 Variabili globali

Per le prime settimane di sviluppo, il progetto è stato realizzato su un unico file sorgente. Ciò ha dato la possibilità di creare variabili globali per il passaggio tra le diverse funzioni. Chiaramente la divisione del progetto in molteplici file sorgenti ha rimosso questa possibilità, e ha portato alla creazione di un tipo definito dall'utente `settings_t`, un contenitore di dati di diversa natura utili alle diverse funzioni; il problema è stato risolto dichiarando e inizializzando una variabile del suddetto tipo e passandone il puntatore alle funzioni.

3.2 Discesa dei tetramini

La prima metodologia di discesa dei tetramini nel campo prevedeva un utilizzo congiunto delle funzioni `t_gravity` e un abbozzo della funzione `t_move`, contenute in *src/tetramino.c*, tramite una strategia di multithreading attraverso la libreria `pthreads`. Tuttavia, implementare questa strategia si è dimostrato essere ben oltre le mie capacità, perciò `t_gravity` è stata messa da parte e riutilizzata successivamente per l'algoritmo di gioco della CPU e `t_move` è stata rifatta completamente in modo da reggere da sola lo spostamento dei tetramini nelle diverse direzioni, compresa la loro caduta.

3.3 Algoritmo CPU

L'adozione di algoritmi specifici per il metodo di gioco della CPU mi è sembrato fin dall'inizio al di fuori delle mie capacità, ma al contempo ero convinto di poter fare di più di una semplice strategia randomica. La soluzione che ho trovato consiste nel far fare al calcolatore due tentativi:

- `cpu_hit` scorre tutti i possibili tetramini con le relative rotazioni in ogni posizione del campo, e se trova una combinazione che riesce a fare punti va a colpo sicuro e la inserisce nel campo.
- `cpu_guess` viene chiamata quanto `cpu_hit` fallisce. Valutando la riga vuota più bassa nel campo, conta il massimo numero di spazi consecutivi nella riga sottostante, e ne calcola i bordi sinistro e destro. Se questo numero è inferiore ad una data soglia, analizza gli spazi circostanti e valuta un pezzo da inserire con una precisa rotazione. Solo quando questo numero di spazi consecutivi è relativamente grande allora la scelta del pezzo è randomica, ma questo verrà comunque piazzato in quell'arco di posizioni.

3.4 Limitazioni ANSI C

Rifinendo i dettagli del progetto nel tentativo di rendere il codice il più possibile attinente allo standard ANSI C, si sono verificate diverse situazioni in cui c'è stato bisogno di riscrivere intere porzioni di codice in modo macchinoso e in molte più righe rispetto alla versione originale (un esempio è la funzione `init_pieces`, che assegna elemento per elemento gli array dei tetramini). La soluzione a questo problema è stata solamente tanta pazienza.

3.5 Difficoltà tecniche: suddivisione in librerie, Makefile...

Nonostante sia stato spiegato a lezione come dividere un progetto in sorgenti e header, nella pratica ho dovuto impegnarmi per comprendere a pieno come creare un progetto pulito e diviso in librerie multifunzionali.
Discorso analogo per il Makefile, che tuttora riconosco non essere un capolavoro di efficienza, ma starci sopra a sperimentare e giocare mi ha dato una discreta infarinatura di come funzionano le macro e come si gestiscono compilazione e linking via Make.

4 Riferimenti

- Gestione input e temporale: per il risultato che ho ottenuto per quanto riguarda lo spostamento dei pezzi nel campo congiunto alla loro discesa devo ringraziare il mio collega Mirco De Zorzi, che si è reso disponibile per presentare a lezione le sue soluzioni in merito. È per gran parte grazie a questo suo contributo che posso dire di essere riuscito a creare alcune funzioni definite in *utilities.c* quali `current`, `raw_enable`, `raw_disable` e `select_char`.
  Inoltre, è sempre da lui che ho preso ispirazione per l'impostazione del file system del progetto.
  Link alla repository del suo progetto: `https://github.com/mircodezorzi/CT0441`
- Funzione `msleep`: `https://qnaplus.com/c-program-to-sleep-in-milliseconds/`