Progetto Bomberman per Programmazione
Università di Bologna

Lorenzo Forese

1 Funzionamento Generale

Il programma inizia dal file `main.cpp`, nel quale vengono inizializzate tutte le classi necessarie al funzionamento del progetto. Ogni classe è dotata dei rispettivi file di intestazione `.h` e di implementazione `.cpp`.

1.1 Classi principali

Le classi richiamate dal programma sono le seguenti:

- Lang
- Finestre
- Selezioni
- Impostazioni
- Leaderboard
- Game
  - mappaRandom
  - movimento
  - AI

2 Lang

La gestione delle lingue avviene tramite un approccio semplice: per ogni opzione testuale è presente un array di stringhe, nel quale ogni posizione corrisponde a una lingua specifica.

Le lingue supportate sono:

- 0 – Inglese
- 1 – Italiano
- 2 – Spagnolo
- 3 – Tedesco

- 4 – Francese

Il cambio della lingua avviene modificando l'indice dell'array, permettendo così l'aggiornamento automatico della lingua in tutte le sezioni del programma che ne fanno uso.

3 Finestre

Questa classe utilizza le funzionalità offerte dalla libreria `ncurses` per creare e gestire le finestre di gioco. Le finestre principali sono:

- `mainwin`: la finestra principale che contiene l'intero programma (più piccola della finestra del terminale, denominata `stdscr`).
- `gameWin`: la finestra che contiene il gioco e le informazioni mostrate all'utente.

La finestra `mainwin` viene inizializzata solamente se le dimensioni del terminale rispettano i requisiti minimi richiesti.

4 Selezioni

La classe `Selezioni` utilizza l'attributo `A_REVERSE` fornito da `ncurses` per rappresentare visivamente l'opzione attualmente selezionata.

Essa si affida alla classe `Lang` per la corretta traduzione delle voci di menu e alla classe `Finestre` per la corretta visualizzazione nella posizione desiderata.

5 Impostazioni

La classe `Impostazioni` utilizza `Lang` per riferimento, così da poter modificare direttamente i valori interni e quindi cambiare la lingua del programma.

Anche questa classe fa riferimento a `Finestre` per la stampa delle informazioni nella posizione corretta.

6 Leaderboard

Questa classe, creata con l'aiuto di un LLM, utilizza la libreria `fstream` per la gestione dei file. I punteggi dei giocatori vengono ordinati in modo decrescente e salvati al termine della partita, secondo le condizioni definite nella funzione `inGame`.

Se il file `leaderboard.txt` non è presente nella directory dalla quale viene avviato l'eseguibile, esso viene creato automaticamente. Utilizzando `cmake` e una directory separata come `/build`, i punteggi precedenti non risultano accessibili. Questo problema può essere risolto in due modi:

- Spostare l'eseguibile o il file `leaderboard.txt` nella stessa directory.
- Modificare il percorso del file in modo appropriato.

La classe è stata successivamente modificata, come descritto nella sezione *Utilizzo LLM*.

7 Game

La classe `Game` contiene la funzione booleana principale `inGame`, che indica se la partita è in corso o terminata.

All'inizio di `inGame` viene definito un valore `float` che rappresenta la difficoltà del gioco, inizializzato a `0.1`. Questo valore influisce su diversi parametri:

- Tempo disponibile: $t - (\text{difficoltà} \cdot t)$
- Numero di muri distruttibili (probabilità di spawn, es. $0.1 = 10\%$)
- Numero di nemici (probabilità di spawn, es. $0.1 = 10\%$)
- Oro guadagnato: $\text{oro} \cdot \text{difficoltà}$

Successivamente vengono inizializzate le classi necessarie:

- `mappaRandom`
- `movimento`
- `Ai`

Infine vengono eseguite ulteriori inizializzazioni, come la gestione del tempo, la posizione di partenza, e la rappresentazione iniziale della mappa.

8 mappaRandom

La classe `mappaRandom` contiene diverse funzioni fondamentali per la stampa della mappa e lo spawn casuale del protagonista. Include inoltre due funzioni di supporto per la generazione di numeri pseudo-casuali.

La funzione principale è `printMappa`, che si occupa di stampare la mappa di gioco e, prima di essa, informazioni quali vite, punteggio, denaro e tempo rimanente.

9 movimento

La classe `movimento` gestisce il movimento del protagonista, la posizione di spawn del giocatore e dei nemici. Si occupa inoltre del piazzamento delle bombe e della gestione degli acquisti nel negozio, verificando che il saldo disponibile sia sufficiente al prezzo dell'oggetto selezionato.

10 Ai

Il gioco prevede due tipologie di nemici:

- KamiKaze
- Bombarolo

10.1 KamiKaze

Il nemico KamiKaze è il più semplice. Effettua controlli di base nel perimetro circostante per verificare la presenza del giocatore. Quando il protagonista si trova a distanza 1, il KamiKaze diventa una bomba al primo tick disponibile.

Il movimento avviene controllando in ordine le celle sottostanti, sovrastanti, a destra e a sinistra. Il primo spazio libero in questo ordine viene scelto per lo spostamento. Dopo il movimento, il nemico diventa `KamiKaze_Disattivo`, per evitare ulteriori spostamenti nello stesso ciclo di controllo.

10.2 Bombarolo

Il nemico Bombarolo presenta una logica più complessa. Appena spawnato, individua la posizione del protagonista e tenta di avvicinarsi prima lungo l'asse delle ascisse. Se la cella più vicina è libera, si sposta in quella posizione; altrimenti esegue lo stesso controllo lungo l'asse delle ordinate.

Quando il Bombarolo si trova a distanza 1 dal protagonista, piazza una bomba e diventa un `Bombarolo Disarmato`. In questo stato, ad ogni ciclo di gioco, tenta di raggiungere un angolo casuale della mappa.

10.3 Utilizzo LLM

Il modello di LLM utilizzato è quello di chat gpt. Dato che per molte funzioni le richieste ritornavano codice con molteplici falle, problemi quando implementate con altre funzioni oppure semplicemente non funzionanti, ho optato per l'utilizzo minimo possibile di codice copiato e incollato dalla LLM al codice direttamente. Ho invece preferito controllare personalmente ogni funzione e scriverla a mano, chiedendo eventualmente alla LLM cose di natura specifica come i caratteri che avrei potuto utilizzare secondo la specifica ASCII. Per il resto mi sono completamente affidato a conoscenze pregresse e/o ottenute durante il corso di programmazione e al sito NCURSES Programming HOWTO per informazioni sulla libreria e sul suo funzionamento.

L'unico file completamente affidato alle LLM è stato il file Leaderboard.cpp (il file header, anche se semplice l'ho completato io) in cui sono presenti due funzioni, ottenute con il seguente prompt e alcuni piccoli accorgimenti sucessivi:

Devo fare una funzione in C++ per il mio progetto di programmazione.

Ho un file leaderboard.txt con dentro nome e punteggio e devo aggiungere un nuovo punteggio alla fine della partita.

La funzione riceve il nome del giocatore come string e il punteggio come int.

Poi deve:

leggere il file

aggiungere il nuovo nome e punteggio

ordinare i punteggi dal più alto al più basso

riscrivere il file aggiornato

Non posso usare vector.

Puoi scrivermi la funzione?