CppMon

Il programma è stato suddiviso nelle seguenti classi:

1. match
2. player
3. monster
4. attacchi
5. Logger
6. randFz

La classe *RandFz* contiene semplicemente tre funzioni che restituiscono valori casuali.

All'avvio del programma vengono create due liste contenenti rispettivamente tutti i mostri e tutti gli attacchi presi dai file *.txt* forniti. Queste liste saranno i "calderoni" di riferimento usati solo in fase di creazione dei team dei due giocatori. Dopo questa fase le due liste vengono eliminate perché non più utilizzate.

La parte appena descritta la si può definire come "inizializzazione" dell'intero programma ed essa è collocata nel *"constructor"* della classe *"match"*.

Una particolarità introdotta è la creazione completamente automatica del team gestito dal programma (NPC - NonPlayer Character) nel caso in cui questa squadra non fosse definita nell'apposito file di testo (cioè se fosse solo definito un giocatore con la rispettiva squadra).

Il "cuore"del programma, dove viene gestito lo scontro, è una funzione membro della classe *match*, chiamata *"fight()".* Questa, collegandosi ad altre funzioni della classe e appoggiandosi anche alle funzioni public degli oggetti che contiene, gestisce un intero combattimento tra due avversari.

Le classi *"match"*, *"player"* e *"monster"* sono legate alla classe *Logger*; ovvero ogniqualvolta un oggetto appartenente ad una di queste classi viene creato, ad esso viene creata e collegata un'"appendice" di tipo *Logger* legata univocamente all'oggetto creato.

Essendo ridotto il numero di attacchi e di mostri fornito, e non essendo la disposizione dei rispettivi *ID* ordinata, non è stato necessario utilizzare algoritmi complessi di ricerca, anzi, risulta più efficace una ricerca sequenziale dal primo all'ultimo elemento.
Nel caso in cui i numeri di questi dati (attacchi e/o mostri) dovessero salire allora potrebbe essere opportuno optare per una ricerca di tipo binario; siccome questi due tipi di dati sono immagazzinati in *vector* si può sfruttare, visto che è già fornita, la libreria *<algorithm>* e usare in ordine le seguenti funzioni:

- `std::sort (v.begin(), v.end());`
- `std::binary_search (v.begin(), v.end(), fz);`

dove **v** è la variabile definita *vector*, mentre **fz** è una funzione che ritorna un valore bool in base al confronto tra due dati (nel nostro caso i dati sarebbero gli *ID* dei mostri o degli attacchi).

Sono state usate molto variabili e funzioni di tipo "bool" perché occupano meno spazio in memoria rispetto agli interi (8bit vs 32bit) e garantiscono maggior velocità in fase di confronto. Quindi quando

occorreva implementare caratteristiche che potevano avere solo due valori, queste sono state tradotte in tipi booleani.
Così come in alcuni casi sono state usate variabili di tipo "char" (8bit).

È previsto che nel caso di aggiunta di ulteriori stringhe nei file *.txt* forniti, queste siano perfettamente conformi agli esempi presentati.
Nonostante ciò, il codice è stato scritto (in certi casi) per avvisare l'utente in caso di possibili errori scritti nei file.