Relazione progetto di programmazione (Gioco RPG)

Alla realizzazione di questo progetto hanno partecipato gli studenti: Lorenzo Girotti **0001020884**, Carolina Bonafè **0001031089** e Rafid Fardeen Hoque **0001027503**.

Lorenzo Girotti si è occupato di gestire le classi **Player** e **Level**, di tutti i metodi che coinvolgono queste classi, inoltre ha gestito il passaggio tra livelli e delle dinamiche che comprendono la gestione del menu di gioco.

Carolina Bonafè si è occupata delle classi **Wall** e **StaticEntity**, quest'ultima utilizzata per la gestione di poteri ed artefatti, di tutti i metodi che coinvolgono queste classi, inoltre ha creato tutti i template dei muri interni utilizzati nei vari livelli.

Rafid Fardeen Hoque si è occupato delle classi **Enemy** e **Bullet**, di tutti i metodi che coinvolgono queste classi, inoltre si è occupato della gestione della stanza segreta e di tutte le dinamiche al suo interno.

Il gioco

IMPORTANTE : **APRIRE IL GIOCO CON IL PROMPT DEI COMANDI A SCHERMO INTERO**

Il gioco RPG si basa su livelli, potenzialmente infiniti, i quali vengono popolati da entità di vario tipo oltre al player, come muri interni, nemici, poteri ed artefatti.

L'obiettivo del gioco è quello di raccogliere tutti gli artefatti presenti nel livello per poter passare al livello successivo, questi artefatti sono presenti sia nella stanza principale del livello, sia nella stanza segreta, accessibile solo dopo aver sconfitto tutti i nemici presenti nella stanza principale.

Nella stanza principale è disponibile un passaggio, sbloccabile dopo aver raccolto tutti i poteri, necessario per progredire di livello e per sbloccare la stanza segreta, il quale contiene 2 artefatti ed un nemico più forte rispetto a quelli disponibili subito appena entrati nel livello.

Nella stanza segreta invece sono presenti 1 potere e 2 artefatti, "protetti" da un tipo di nemico, identificato col simbolo K, che toglie 15 punti vita quando si trova nella stessa

posizione del player.

Per muovere il player bisogna utilizzare le freccette `Up` `Down` `Left` `Right`, mentre affinché spari bisogna premere la barra spaziatrice `Space` e subito dopo una freccetta per decidere in che direzione sparare il proiettile.

La difficoltà del gioco è data anche dai seguenti punti:

- Il danno dato dai proiettili sparati dal player è invariato per tutta la durata del gioco, ovvero 10 punti vita.
- Il numero di nemici presenti nella stanza principale di ogni livello varia in modo logaritmico rispetto al numero del livello, $\log_2(\text{levelNumber}) + 2$.

  (Es. Lvl. 1 $\Rightarrow$ 3 Enemies . . . Lvl. 4 $\Rightarrow$ 4 Enemies . . . Lvl. 8 $\Rightarrow$ 5 Enemies)
- La vita e il danno provocato dai proiettili dei nemici seguono le seguenti formule:

  Punti vita $\Rightarrow (\log_2(\text{levelNumber}) + 2) * \text{levelNumber}$

  Danno dei proiettili $\Rightarrow (\log_2(\text{levelNumber}) + 1) * 2$

  NB: Il nemico nella stanza segreta e nel passaggio sbloccato dai poteri avrà vita doppia rispetto a quelli nella stanza principale.
- Gli artefatti della stanza principale restituiscono un valore di punti vita pari al danno che infligge un proiettile del nemico in quel livello, mentre gli artefatti della stanza segreta e del passaggio sbloccato tramite i poteri restituiscono il doppio degli artefatti normali.

I livelli

I livelli vengono implementati tramite una lista bidirezionale che contiene varie informazioni, tra cui: il numero del livello, una lista di artefatti, una di poteri, una di nemici e una di muri interni.

La classe Level inoltre contiene vari metodi utili alla creazione e alla gestione del livello, tra i tanti metodi possiamo trovare:

- *initializeLevel*: questo metodo serve per inizializzare il livello per la prima volta con tutte le entità al suo interno a 360 gradi, qui infatti vengono generati tutti i nemici, i

poteri e gli artefatti del livello, le cui posizioni all'interno della mappa vengono scelte in modo randomico.

- *printEntities*: tramite questa funzione vengono stampate tutte le entità del livello contenute nelle varie liste, questo metodo viene richiamato ad ogni ciclo del while utilizzato nel main durante l'esecuzione del gioco.
- *checkCollisions*: questo metodo viene utilizzato per verificare tutte le collisioni che possono esserci all'interno del livello durante l'esecuzione del gioco, ovvero del player sopra gli artefatti e poteri e dei proiettili del player sui nemici e viceversa.

Le entità

Le entità possono essere di vario tipo e tutte hanno in comune il fatto di possedere delle coordinate e un carattere che verrà stampato sulla mappa (Es. @ per il Player, E per i nemici, A per gli artefatti e P per i poteri).

Inoltre, tutte le entità possiederanno un flag che segnala se queste devono essere stampate nella stanza principale del livello oppure nella stanza segreta.

Queste informazioni comuni a tutte le entità sono racchiuse nella classe Entity, classe padre di tutte le altre entità.

StaticEntity

Questa classe serve per definire i metodi utilizzati durante la generazione dei poteri e degli artefatti, inoltre è presente un metodo che serve a verificare se durante la generazione dell'entità, è possibile occupare la posizione scelta precedentemente in modo randomico.

Bullet

Nella classe Bullet vengono definiti 2 attributi in più rispetto a quelli già presenti in Entity: uno per il danno ed uno per definire l'offset del proiettile, il quale verrà sommato alle sue coordinate per aggiornarne la posizione.

I metodi di questa classe sono tutti relativi alla gestione e alla stampa del proiettile all'interno della mappa.

LivingEntity

Le LivingEntity, che nel nostro caso possono essere il player oppure i nemici, sono un tipo di entità che differisce dalle altre perché possiede una lista di proiettili, questo perché solo le LivingEntity hanno la possibilità di sparare proiettili.

Potendo anche spostarsi, in questa classe vengono definiti tutti i metodi utili allo spostamento nella mappa e, come detto in precedenza, per gestire la lista di proiettili.

Player

Questa classe, sottoclasse di LivingEntity, è stata creata per gestire le particolarità del Player rispetto ad una normale LivingEntity, qui infatti sono presenti i metodi utili al transito tra livelli da parte del Player e i metodi utilizzati per "resettare" il Player in caso venga scelto di iniziare una nuova partita.

Enemy

La classe Enemy è una sottoclasse di LivingEntity, è stata creata per gestire le funzioni aggiuntive che ha il nemico rispetto ad una normale LivingEntity come il Player.

Qui viene gestita tutta la parte di AI del nemico, la quale regola gli spostamenti e la decisione di sparare da parte del nemico, che nel caso del nemico nella stanza segreta sarà nulla non potendo sparare.

Inoltre, come nella classe StaticEntity, è stato creato un metodo che serve a verificare se, durante la generazione dell'entità, è possibile occupare la posizione scelta precedentemente in modo randomico.

I muri interni (Wall)

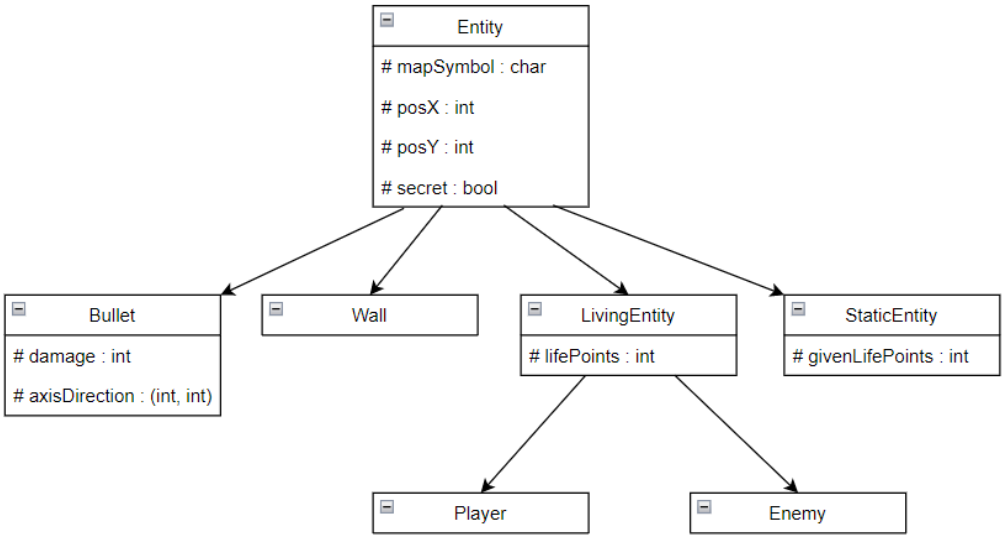
Per i muri interni sono stati realizzati 4 template di muri che si ripetono ogni 4 livelli.

Ogni template consiste in una lista semplice di muri (oggetti di tipo Wall), la quale viene utilizzata per:

- regolare tutti gli spostamenti delle entità (le LivingEntity) all'interno della mappa;
- vincolo per la validità dei proiettili, i quali verranno eliminati quando si scontreranno contro i muri interni;
- i controlli che regolano in che posizione possono essere generati gli artefatti, i poteri ed i nemici quando viene inizializzato un nuovo livello.

L'ereditarietà nelle entità

Questo schema mostra com'è stata gestita l'ereditarietà tra le entità durante lo svolgimento del progetto.



Le costanti

Durante la creazione del progetto è stata creata una libreria chiamata **constants**, la quale contiene tutte le costanti utilizzate nel programma, tutti i metodi utilizzati per gestire la stampa delle WINDOW di ncurses e per le altre parti dell'esecuzione del gioco più statiche e/o grafiche.

Grazie per l'attenzione.