Presentazione: Progetto di programmazione

Limiao Jin, Jinxiufu Lin, Lorenzo Giordani, Claudio Jin

January 17, 2022

Descrizione del gioco

Il gioco è un semplicissimo RPG in grafica ASCII. Il giocatore esplora diversi livelli e mappe e cerca di sopravvivere ai nemici mentre accumula punti e vite. La difficoltà cresce di livello in livello: aumenta il numero di nemici, il loro movimento e la loro forza di pari passo con il numero degli artefatti. Il gioco termina quando il giocatore esaurisce le vite oppure quando viene premuto il tasto 'q'. Sono presenti diversi tipi di nemici, artefatti e mappe

Implementazione

L'implementazione è basata sulla definizione di classi (Entity, Monster, Player) che interagiscono attraverso funzioni e strutture (Struct, Level, Graphics, Physics).

Classi

Entity:

Entity è la classe padre che contiene le informazioni fondamentali di tutto quello che è presente sulla mappa. I suoi attributi sono:

- `position`: coordinate della posizione dell'entity
- `texture`: carattere ASCII che rappresenta l'entity sulla mappa
- `Active`: variabile booleana che indica lo stato di attività

I suoi metodi oltre ai metodi get e update per ottenimento e modifica diretta degli attributi sono:

- il costruttore, che inizializza opportunamente gli attributi
- `is_move_ok()`: stabilisce la validità della posizione corrente
- `death()`: rende l'entità inattiva

Monster:

Monster è la classe figlio di Entity con gli attributi aggiuntivi:

- `newposition` : coordinate della posizione successiva dell'entity
- `life`: vita del mostro

E nuovi metodi:

- `damaged()`: decrementa la vita di una unità fino a chiamare il metodo death()

Player:

Player è la classe che definisce il giocatore ed è la classe figlio di Monster. I nuovi attributi sono:

- `score`: punteggio del giocatore
- `bullet`: entity che rappresenta il proiettile del player; ogni proiettile infligge una unità di danno ai nemici

con metodi:

- `scored()`: aumenta il punteggio quando un nemico viene sconfitto
- `lifeUP()`: aumenta l'attributo life di una unità
- `damaged()` : decrementa la vita del player a seconda del mostro con cui è entrato in contatto

Interazioni

Level

I livelli sono organizzati in una lista bidirezionale: ogni nodo è una struct in cui sono contenuti i seguenti elementi:

- `num_lev` è il numero del livello
- `id_map` stabilisce quale mappa disegnare
- `n_monsters` e `n_bonus` indicano il numero di mostri e artefatti presenti nel livello
- `EntrDoor` ed `ExtDoor` rappresentano le coordinate y delle porte di entrata e uscita
- `tp_flag` e `Show_tp` sono variabili flag per i poteri di teletrasporto, per la presenza e la visualizzazione
- `DoorOpened` indica se la porta è aperta o meno

Graphics

Definisce le funzioni necessarie per mostrare la mappa

Physics

Contiene funzioni che:

- Regolano il movimento:
  - `controll()`: funzione per controllare il player
  - `UpdateMonsterposition()`: aggiorna la posizione dei nemici; random nei primi livelli, verso il player nei livelli successivi
- Aggiornano la posizione dell'entità:
  - `updateMonster(), updateBonus(), updatePlayer()`
  - `CheckPosition()` : controlla la validità della posizione iniziale; se la posizione non è valida, ne assegna una nuova
  - `CheckDoor()` controlla se tutti i nemici sono morti e quindi apre la porta d'uscita
- Stabiliscono i contatti tra le entità:
  - `bullet_hit()`
  - `Player_hit()`: funzione booleana che stabilisce se il player entra in collisione con altre entità
  - `Hitbox()`: stabilisce gli effetti dei contatti tra le entità

Suddivisione del lavoro

Il lavoro è stato diviso tra i componenti del gruppo nel seguente modo:

- Lin : Base del lavoro e coordinazione delle varie parti
- Jin C. : Effetti dei contatti e altre funzioni collegate
- Giordani: Definizione e aggiornamenti delle varie strutture e classi
- Jin L.: Grafica e altre funzioni collegate

La suddivisione è indicativa e le parti sono state svolte in collaborazione di tutti gli elementi del gruppo