*Marius Grigoroiu | Informatica 23/24 A-F*

**JSimpleSMT**

Esplorazione di dungeon generati per piani casualmente con visuale in prima persona (ref.: RPG vecchia scuola, Shin Megami Tensei / The Bard’s Tale [[link](https://tcrf.net/images/0/0e/Smt1_severe_haze.png)][[link](https://gaming-cdn.com/images/products/12252/screenshot/the-bard-s-tale-trilogy-xbox-one-xbox-series-x-s-xbox-series-x-s-xbox-one-gioco-microsoft-store-europe-wallpaper-2.jpg?v=1703167739)]).

**Generazione piani**

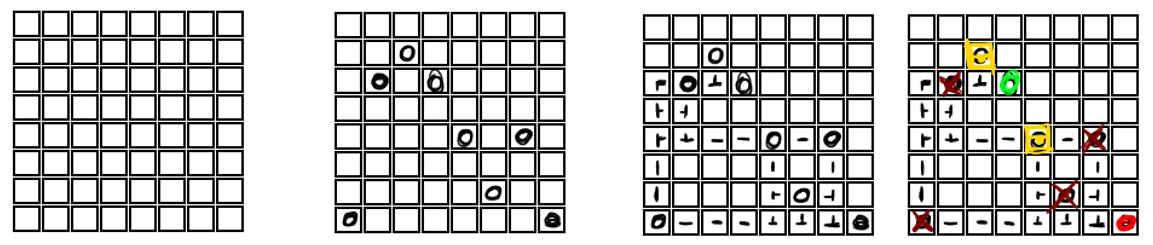
In una griglia AxB verranno generati un numero X (radice arrotondata per difetto di AxB) di punti d’interesse in posizioni random della griglia, punti che verranno poi collegati da dei corridoi. Quindi ogni casella potrà essere una *stanza (punto d’interesse)*, un *corridoio*, o *non interagibile.*

Un punto d’interesse dovrà contenere un *nemico*, una *cassa* (recupero HP, guadagno di punti esperienza), oppure un *punto d’entrata/uscita* per il piano.

Approfondendo l’idea, a lato codice, una stanza con punto di riferimento sarà uguale una “casella” corridoio, considerando però il fatto che una sarà interagibile mentre l’altra no.

Punti di riferimento:

* Entrata/Uscita per il piano
* Nemico
* Baule/ricompensa
  + Punti esperienza
  + Punti vita
* Trappole (perdita automatica di PV)



**Statistiche personaggio**

Il *personaggio controllabile* ed i *nemici* avranno in comune le statistiche di:

* *attacco*
* *difesa*
* *punti salute*
* (i nemici avranno anche un numero di punti esperienza che verrà concessa al giocatore dopo averli uccisi)

Mentre il personaggio controllabile comprenderà anche:

* tassi di crescita in percentuale delle statistiche
* livello
* punti esperienza
* punti esperienza necessari per un aumento di livello

**Sistema di combattimento**

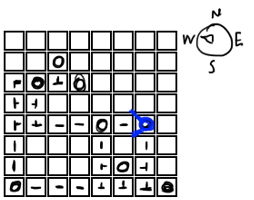
Il sistema di combattimento verrà istanziato nel momento in cui il personaggio sarà nella stessa casella in cui sarà presente un nemico, interrompendo la fase di esplorazione.

Per cercare di mantenere il livello di complessità in linea con il progetto originale “JPokeBattle”, ho pensato fosse superfluo un sistema di mosse/skill, limitando quindi l’eventuale interazione tra personaggio giocabile e nemico ad un “infliggi danni” (attacca), “difenditi” (dimezza i danni subiti) o fuggi.

Azioni disponibili:

* attacca
  + togli PS (ATK dell’attaccante - DEF del nemico) al personaggio con cui stai interagendo
* difenditi
  + salta il turno e raddoppia la statistica della difesa per un turno
* fuggi

Portando i PS del nemico a 0 (come ricompensa un aumento dei punti esperienza) o fuggendo (probabilità rapportata al proprio livello e quello del nemico), il programma ritornerà automaticamente al sistema di esplorazione.



**Movimento personaggio**

Ad ogni ciclo di movimento durante l’esplorazione verranno salvati la *posizione* del personaggio nella griglia e la *direzione* verso la quale si sta rivolgendo (NORD, OVEST, SUD, EST), così da potersi:

* Muovere in avanti (verso la direzione rivolta)
* Ruotare di 90° verso destra o sinistra (cambiare direzione)

Quindi se il personaggio è rivolto verso EST:

* Muovendosi in avanti si sposterebbe di una casella verso EST (se la casella è un corridoio o una stanza con punto di riferimento)
* Ruotando verso sinistra la visuale si sposterebbe verso NORD
* Ruotando verso destra si sposterebbe verso SUD

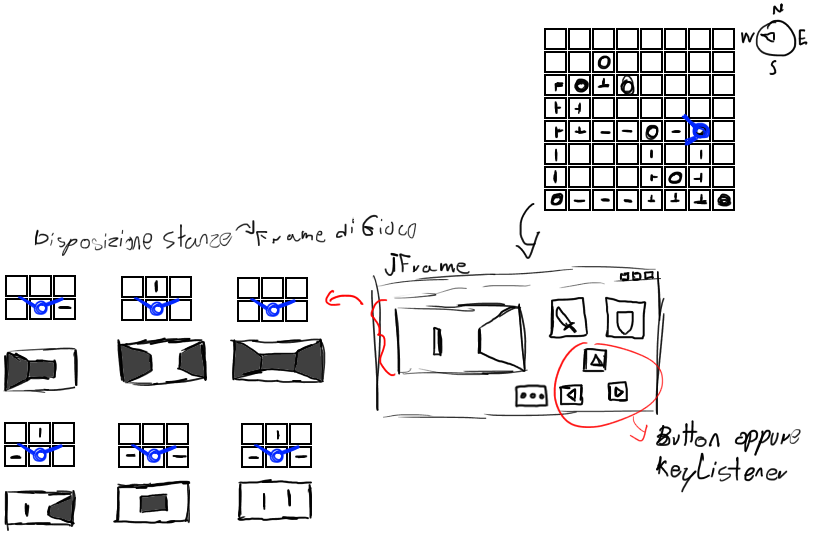
Sia il sistema di combattimento che quello di esplorazione chiaramente non saranno in tempo reale, quindi ogni azione corrisponde ad un input da parte del giocatore, mentre il gioco aspetterà una risposta.

**Implementazione con Swing**

La visuale in prima persona accennata prima sarà implementata tramite un sistema di immagini: ogni immagine verrà associata a ciò che il “Playable Character” sarà in grado di vedere ed a ciò che gli succederà intorno. Ogni azione sarà associata ad un JButton, incluse le opzioni di movimento

Le immagini saranno pre-disegnate/pre-renderizzate in prospettiva centrale *andando a visualizzare esclusivamente la casella frontale e le due laterali*: un’immagine per ogni disposizione delle mura attorno al personaggio.

Per esempio: se il personaggio avrà davanti a se’ un corridoio/stanza, ma niente ai lati, verrà visualizzata un’immagine rappresentante due muri ai lati e una strada libera, ignorando qualsiasi altra casella (9 immagini totali) [veda secondo esempio nell’illustrazione].



*In via alternativa, un altro modo di rappresentazione (fallback) può essere quello delle 8 caselle attorno al giocatore tramite visuale dall’alto.*



**Priorità**

Essenziali:

* Algoritmo di generazione piani
* Statistiche personaggio / nemici
* Aumento di livello / punti esperienza
* Sistema di esplorazione / combattimento
* Implementazione con Swing

Utile:

* Visuale in prima persona (non renderizzata in tempo reale)
* Casse / ricompense
* Adattabilità dell’algoritmo di generazione del piano a seconda della progressione della partita (difficoltà proporzionale al numero del piano):
  + Dimensione piano
  + Numero di mostri
  + Potenza mostri
* KeyListener

Extra (improbabile):

* Trappole
* Più tipi di nemici
* Boss
* Animazioni nei combattimenti ed effetti sonori
* File .jar
* JUnit
* Interazione opzionale (tramite tasto) con i punti di riferimento