Relazione del gioco

Andrea Berlingieri, Giacomo Puligheddu, Riccardo Bellelli

Indice

1 Il gioco **2**

2 Scelte implementative **3**
2.1 La mappa 3
2.2 Livelli 3
2.3 Grafica 4
2.4 La battaglia 4
2.5 I personaggi 5
2.6 Gli oggetti 6
2.7 Inventario 6
2.8 I mostri 6
2.9 Lo Shop 7

Capitolo 1

Il gioco

Il gioco è un roguelike molto semplice nelle sue meccaniche. Il giocatore esplora un dungeon formato da livelli generati in modo procedurale, combattendo con vari mostri con un sistema a turni. L'obiettivo del gioco è la raccolta di un numero prefissato di oggetti speciali. Nello specifico, è possibile scegliere tra tre personaggi già definiti, ognuno con la sua backstory e le sue statistiche, che devono esplorare il dungeon alla ricerca dei 12 pezzi di laurea necessari ad uscire dal luogo misterioso in cui il player si trova. I comandi del gioco sono molto semplici e sono accessibili in-game con il tasto '?'.

Nota: ogni riferimento a cose e persone reali è puramente casuale. Il gioco ha uno sfondo puramente satirico.

Capitolo 2

Scelte implementative

2.1 La mappa

La mappa è generata proceduralmente. Si parte da un'area iniziale che viene divisa in due, dopodichè si dividono le due aree ottenute, poi le quattro ottenute ora, e così via, fino a raggiungere un numero di aree che non si intersecano pari al numero di stanze richieste. Quando il numero delle stanze diventa grande è possibile che l'algoritmo di divisione non riesca a terminare con un numero di aree uguale a quello richiesto. Per questo motivo all'algoritmo sono permessi un numero fisso di tentativi. La dimensione delle aree dopo la suddivisione è decisa casualmente, tenendo conto che la divisione può creare aree la cui dimensione minima delle sottoaree è un terzo dell'area iniziale. In ognuna di queste aree viene generata una stanza interna all'area.

Per la generazione di una stanza si sceglie casualmente un punto all'interno dell'area tale da determinare una stanza di dimensioni minime 5x5 e dimensioni massime 20x20, ammesso che l'area che contiene la stanza sia sufficentemente grande. Successivamente sono scelte in modo casuale, ma rispettando i limiti sulle dimensioni, l'altezza e la larghezza della stanza.

Dopo che le stanze sono state generate queste vengono collegate mediante corridoi. Un corridoio di collegamento tra una stanza e l'altra è ottenuto mediante la ricerca di un cammino minimo tra un punto (casuale) della prima stanza ed uno dell'altra in un grafo che contiene tutti i punti dell'area iniziale meno il bordo e le stanze.

Una volta che le stanze e i corridoi sono stati generati vengono piazzati sulla mappa, che mantiene una griglia di tiles inizializzate a WALL.

2.2 Livelli

Un livello è definito dal suo numero, dalla sua mappa, dai suoi mostri e dai suoi oggetti. Questi sono mantenuti in delle tabelle hash che come chiave utilizzano l'id dell'oggetto o mostro. In questo modo ogni volta che si accede allo stesso

livello la mappa, la disposizione dei mostri e degli oggetti rimangono uguali a quando si abbandona il livello.

Per passare da un livello all'altro si utilizzano delle scale. Si hanno sia scale per salire di livello, che scale per tornare indietro di un livello, fatta ovviamente eccezione per il primo.

I livelli vengono mantenuti in una lista ed il passaggio da un livello ad un altro corrisponde all'incremento o decremento di un iteratore che punta al livello corrente. Se la lista "finisce" un nuovo livello viene creato.

2.3 Grafica

Per la parte grafica del gioco si è utilizzata la libreria ncurses.h insieme alle librerie aggiuntive menu.h e panel.h, che definiscono alcuni "widget" per ncurses che ne rendono l'utilizzo pratico più agevole. Le strutture dati che rappresentano le finestre ed i menù in ncurses sono state "racchiuse" in dei wrapper ad oggetti. In questo modo la creazione di una finestra o di un menù è resa più agevole tramite l'utilizzo di appositi costruttori, e una volta che la loro "vita utile" termina la memoria occupata viene liberata automaticamente tramite appositi distruttori. Inoltre le azioni di uso comune (come la stampa di una linea, la pulizia di una finestra, la scelta di un'opzione di un menu, ecc.) sono eseguite mediante in maniera semplice apposite funzioni membro.

2.4 La battaglia

Lo svolgimento della battaglia è mostrato al di sotto della mappa, in una finestra nella quale appaiono le statistiche del nemico e le perdite durate gli attacchi da parte di entrambi gli avversari, tenendo aggiornati i parametri del player e del mostro. Inoltre se viene scelto l'inventario verrà aggiornata la finestra degli equipaggiamenti così da permette al giocatore di visionare che parti dell'armatura indossa.

L'inizio della battaglia è caratterizzato dall'attacco del personaggio principale, il quale può scegliere tra tre differenti opzioni:

1. **Attacco**: il player attacca l'avversario, può scegliere tra due diverse mosse:
    - La prima è l'attacco normale, che toglie all'avversario punti vita in base all'attacco del player e alla difesa del nemico, ed in questo caso è presente una scelta randomica, che determina se l'attacco sarà oppure no critico; se così fosse, il PG ottiene attacco raddoppiato, aumentando il danno inflitto al mostro;
    - La seconda riguarda la mossa speciale, che differisce in base al personaggio: Gaudenzio sfrutta la rigenerazione (guadagna LP in base al suo livello), Peppino utilizza un attacco magico (attraverso i punti mana raddoppia l'attacco) e Badore sfrutta l'attacco furtivo (può

triplicare i danni al nemico, ma se viene scoperto perde il turno e il mostro fa due attacchi).

Se non viene scelta nessuna delle due opzioni, il player perde un turno per aver sbagliato a digitare, lasciando l'attacco all'avversario.

2. **Inventario**: si può consultare l'inventario e guardare gli oggetti, si possono equipaggiare, disequipaggiare, droppare o utilizzare se sono consumabili. Tuttavia si perderà un turno di gioco, consentendo al nemico di attaccare.

3. **Corruzione**: invece che combattere il nemico, il player può affidarsi a una scelta randomica e e nel qual caso vada a buon fine, può far si che il mostro se ne vada, pagando un determinato numero di monete (Cucuzze).

Il mostro possiede solo l'attacco normale e il suo colpo potrebbe fallire, mediante una scelta randomica. La battaglia tra player e avversario prosegue finché uno dei due non riesce a sopraffare l'altro. Se il player vince, guadagna una somma di denaro che dipenderà dal tipo di mostro affrontato. Nel caso opposto, cioè se vince il nemico, il gioco finisce in quanto si è stati sconfitti.

2.5 I personaggi

La classe personaggi contiene tutte le statistiche relative al PG, compreso il nome e la posizione sulla mappa. Per quanto riguarda l'inventario, questo fa parte della classe del pg ed è implementato come un unordered set, ovvero come un insieme di item senza un ordine particolare. Tale scelta consente di avere dei tempi di accesso agli item molto bassi rispetto a un'altra struttura dati. Oltre all'inventario è presente un vettore chiamato equipaggiamento che contiene, appunto, gli oggetti che il pg ha equipaggiato.

Oltre ai metodi set, get, pickItem, dropItem.. questa classe contiene anche un metodo LVLup che si occupa di far salire le statistiche del PG, in relazione al PG scelto: il guerriero non otterrà mana e avrà un incremento di punti vita mentre il mago avrà più mana e meno punti vita; il ladro otterrà una maggiore fortuna. I personaggi salgono di livello man mano che si scoprono nuovi livelli: se ci troviamo per la prima volta al livello n, il PG passerà da livello n-1 a livello n. Questo permette un bilanciamento tra le statistiche del PG e quelle dei mostri. Il metodo showInventory è all'interno della classe personaggi perchè questo ha contribuito a una realizzazione più semplice (dato che così può accedere ai campi del PG). Tale metodo gestisce l'interazione dell'utente con l'inventario del suo personaggio. Infine si è rivelato utile utilizzare una funzione (suffix) per poter decidere come riferirsi a un determinato oggetto: alcuni item sono maschili, altri femminili e altri al plurale (stivali, spada,..).

2.6 Gli oggetti

La classe Item ha al suo interno tutti i campi relativi alle statistiche modificate, alla posizione sulla mappa, al simbolo con la quale verrà visualizzato e al fatto se l'item è visibile o meno.

Due sono i campi fondamentali: id e type. Il campo id consente di identificare univocamente un Item, ovvero di distinguere più oggetti tra loro, anche con lo stesso nome. Questa scelta è stata necessaria perchè potrebbero comparire più oggetti identici sulla mappa. Per quanto riguarda il campo type, questo è l'indice con la quali gli item verranno inseriti nel vettore equipaggiamento; un item di tipo elmo avrà type uguale a 0, un item di tipo corazza avrà type uguale a 1 e così via. I consumabili sono di type 5 (il vettore equipaggiamento arriva fino all'indice 4).

È stato necessario confrontare gli oggetti tra loro, pertanto sono stati creati due metodi per poter decidere se due item sono uguali o diversi; sono uguali se hanno lo stesso id, diversi altrimenti.

2.7 Inventario

Per rappresentare l'inventario viene creata una finestra contenente un menù che permette di spostarsi tra i vari Item; in una finestra laterale vengono mostrate le statistiche e al di sotto di queste ci sono le opzioni disponibili.

L'inventario si può consultare sia durante la perlustrazione della mappa, sia quando si sta combattendo contro un mostro. Se l'oggetto è un consumabile, una volta usato viene cancellato dall'inventario; Se l'oggetto è equipaggiabile, può essere indossato o rimosso. Gli spazi che non contengono oggetti sono vuoti e vengono contrassegnati dalla scritta "-VUOTO-" così da poter monitorare le capacità dell'inventario.

Per tutti gli oggetti è inoltre disponibile l'opzione di drop, che consiste nel lasciare l'oggetto, liberando così uno slot.

2.8 I mostri

La classe Monster contiene i campi relativi alle statistiche, al livello, alla posizione e al nome. Ha poi altri campi tra cui id (che funziona come per gli item) e symbol che, in base al mostro, indica il carattere con cui quel nemico verrà stampato sulla mappa.

Anche qui sono presenti i vari metodi set e get e i metodi necessari per confrontare due mostri e vedere se sono uguali oppure no (in base all'id).

Il metodo moveMonster gestisce lo spostamento dei mostri sulla mappa. Prevede due casi: casuale oppure inseguimento del PG (se nella stessa stanza). Quando il PG lascia la stanza i mostri iniziano a muoversi casualmente, senza uscire da questa. La scelta è stata necessaria perchè altrimenti, il PG sarebbe sempre costretto ad affrontare i mostri che lo bloccherebbero nei corridoi.

2.9 Lo Shop

Ogni volta che si sale di livello il giocatore ha la possibilità di acquistare uno tra tre oggetti per il PG. Una volta comprato un oggetto (o dichiarato lo stato di povertà) il menù si chiude e il nuovo livello viene stampato. Affinchè non vengano presentati tre oggetti identici, è stata utilizzata una funzione (generateKPermutation) per generare una disposizione di 3 oggetti. Per garantire un buon bilanciamento del gioco, i tre oggetti hanno prezzo relativo alla loro rarità, ovvero all'indice con cui sono stati inseriti nel vettore ItemsSet dalla funzione retrieveItems.