Relazione del gioco

Andrea Berlingieri, Giacomo Puligheddu, Riccardo Bellelli

Indice

Capitolo 1

Scelte implementative

1.1 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 è un terzo dell'area iniziale e la cui dimensione massima è due terzi 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 area massima uguale a quella dell'area di appartenenza meno il bordo e di dimensioni minime 5x5. Successivamente sono scelte, in modo casuale ma rispettando i limiti sulle dimensioni, l'altezza e la larghezza della stanza. Le dimensioni massime di una stanza sono 20x20.

Dopo che le stanze sono state generate, queste vengono collegate mediante corridoi. Un corridoio di collegamento tra una stanza e l'altra è generato 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 i punti delle 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.

1.2 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, passando ad uno superiore o inferiore con le scale.

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 livello.

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.

1.3 Grafica

Per la parte grafica del gioco si è utilizzata la libreria ncurses insieme alle librerie aggiuntive menu e panel, 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 apposite funzioni membro.

1.4 La battaglia

L'inizio della battaglia è caratterizzato da una prima scelta randomica, che decide se il nemico è sufficientemente fortunato da attaccare per primo, nel qual caso questa condizione si avveri il personaggio principale perde il turno e di conseguenza si passa alla mano successiva, nella quale inizia per primo il player, 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'aversario punti vita in base all'attaco del player e alla difesa del nemico, ed anche in questo caso è presente una scelta randomica, che determina se l'attacco possa o non possa essere critico, così da moltiplicare per due l'attacco del personaggio principale e di conseguenza aumentando il danno al mostro; la seconda riguarda la mossa speciale, che differesce in base al personaggio, infatti Gaudenzio sfrutta la rigenerazione, Peppino l'attacco magico che gli raddoppia l'at-

tacco ed infine Badore che sfrutta l'attacco furtivo per triplicare i danni al nemico, ma che se viene scoperto perde il turno senza poter attaccare.

2. **Inventario**: si può consultare l'inventario e guardare quali oggetti sono stati raccolti durante il percorso nel gioco, si possono equipaggiare, disequipaggiare pezzi di armatura e utilizzare i consumabili.

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 (Cuccuzze).

Nel caso 1 la battaglia tra player e avversario prosegue finchè uno dei due non riesce a sopraffare l'altro e quindi azzerargli i punti vita, nel qual caso sia il player a vincere, egli guadagna una somma di denaro che potrà sfruttare nello shop del gioco, oppure per corrompere il nemico. Nel caso opposto, cioè in cui sia il nemico a vincere il gioco finisce automaticamente in quanto si è stati sconfitti.

Nel caso 2, invece, si ha solo una fase transitoria in cui il personaggio cerca di equipaggiarsi o rafforzarsi per affrontare, con maggiore probabilità di successo, il nemico. Con questa scelta però si perde la possibilità di attaccare e si lascia l'attacco all'avversario.

Nel caso 3, si può sfuggiare alla battaglia in qualsiasi momento, sempre che siano soddisfatte le condizioni citate nel punto 3. 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'inventario è un'opzione che può essere attivata sia durante la perlustrazione della mappa, sia quando si sta combattendo contro un mostro. Gli oggetti che si possono salvare al suo interno sono di due tipi: consumabili, quindi che una volta usati vengono cancellati dall'inventario perchè hanno un solo utilizzo che modifica le statistiche del personaggio principale, mentre gli altri oggetti sono di tipo equipaggiamento di conseguenza possono essere equipaggiati oppure disequipaggiati nel qual caso il player li abbia addosso come parte dell'armatura. Gli spazi, invece, che non contengono oggetti sono vuoti e vengono contrassegnati dalla scritta "-VUOTO-" così da poter monitorare le capacità dell'inventario e degli spazi restanti. Per tutti gli oggeti è inoltre disponibile l'opzione di drop, che consiste nel lasciare l'oggetto e poter liberare spazio necessario per un altro oggetto.

Per rappresentare l'inventario viene creata una finestra sulla mappa che permette di spostarsi tra i vari manufatti e permette di scegliere quale sia il più adatto allo scopo del giocatore grazie alla presenza di una finestra lateralmente alla lista, nella quale sono mostrate le caratteristiche di ognuno dei manufatti con al di sotto le opzioni sfruttabili, che cambiano rispetto al tipo di oggetto che si sta guardando in quel momento.

1.5 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. La scelta di un vettore piuttosto che di una lista o una qualsiasi altra struttura, è dovuta al fatto che in tal modo si possono distinguere (usando l'indice del vettore) item dello stesso tipo: 0 indica l'elmo, 1 la corazza, 2 la spada e così via; questo significa che solo un elmo può occupare un posto nel vettore, solo una corazza e via dicendo. Gli Item consumabili, ovvero quelli che una volta utilizzati vengono rimosssi dall'inventario, non sono equipaggiabili e pertanto, dato che il vettore equipaggiamento ha dimensione pari a 5, i consumabili hanno indice/tipo pari a 5.

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. Da notare che è stato creato un apposito metodo per accedere al vettore equipaggiamento; ciò si è reso necessario perchè chiaramente non è possibile restituire un vettore per intero, quindi, dato un indice, viene restituito l'elemento presente in quell'indice del vettore equipaggiamento. 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,..).

1.6 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 sulla mappa.

Due sono i campi fondamentali: il campo id e il campo type. Il campo id consente di identificare univocamente un Item, ovvero di distinguere più item tra loro, anche con lo stesso nome. È stato necessario implementare tale campo perchè potrebbero comparire più oggetti identici nella mappa. Per quanto riguarda il campo type, questo è l'indice con la quali gli item verranno inseriti

nel vettore equipaggiamento; come detto prima, 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 item tra loro, pertanto sono stati creati due metodi per poter decidere se due item sono uguali o diversi; due item sono uguali se hanno lo stesso id, diversi altrimenti.

1.7 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. È importante anche il campo awake, true se il mostro è sveglio e false altrimenti. Questo campo è utile per far sì che il mostro stia fermo fino a quando la stanza in cui si trova non viene scoperta dal PG.

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).

1.7.1 moveMonster

Questo metodo consente lo spostamento dei mostri sulla mappa. Prevede due tipi di spostamenti per i mostri: casuale oppure inseguimento del PG. 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 pedinerebbero lungo tutti i corridoi.

1.7.2 writeInfo

Tale metodo è utile per dare formazioni fondamentali al giocatore: grazie alla finestra generata, l'utente conosce le statistiche del pg, il livello in cui si trova e quanto manca al raggiungimento dell'obbiettivo che gli consente di vincere il gioco. La finestra inoltre ricorda qual è l'abilità del PG scelto all'inizio.

1.7.3 writeEquipment

Il funzionamento è simile a writeInfo ma stampa a schermo gli oggetti equipaggiati. Nell'implementare tale metodo è stato tenuto conto anche del suffisso richiesto dai vari oggetti, tramite la funzione suffix.

1.7.4 shopMenu

Ogni volta che si sale di livello il giocatore ha la possibilità di acquistare uno tra tre oggetti per il PG, consumabili o equipaggiabili. Per non creare confusione nella schermata di gioco, il menù dello shop compare dove dovrebbe comparire la mappa e 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 item 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: quando vengono presi dal file, tramite la funzione retrieveItems, vengono inseriti ordinatamente in un vettore. L'ordine è stato deciso in base alla rarità: rarità e indice del vettore sono direttamente proporzionali, consentendo di generare il prezzo relativo all'item sulla base del suo indice.