Discussione 15/09/2023 - Ivan De Simone, Nicolò Tambini, Tommaso Merighi

Ajeje’s adventures

Progetto Programmazione 2022-23

1 - Introduzione

Ajeje’s adventures è un gioco platform in grafica ASCII realizzato in C++ sfruttando le librerie ncurses. L’obiettivo del gioco è uccidere i nemici per poter avanzare di livello. È un gioco a punti in cui non esistono traguardi, game over se la vita scende a zero. Quando il protagonista muore mantiene armi, potenziamenti e denaro ma perde tutti i punti. Il giocatore può uscire dalla partita in qualunque momento ed i suoi progressi verranno salvati automaticamente.

2 - Gioco

2.1 Protagonista

Il protagonista è rappresentato sulla mappa dal simbolo ‘@’. Esso si può muovere orizzontalmente, può saltare e può infliggere danno ai nemici attaccandoli.
Lo scopo del protagonista è di uccidere i nemici livello dopo livello per accumulare punti e denaro, cercando di perdere meno vita possibile.
L’inventario del protagonista è composto da un’arma ed un potenziamento, che possono essere cambiati nel negozio spendendo il denaro.

2.2 Armi e potenziamenti

Nel gioco sono presenti diversi tipi di arma ed ognuna ha delle caratteristiche che la rendono speciale.

- Bastone: infligge danno nella casella di fronte al protagonista, è l’arma di default all’inizio del gioco.
- Spada: causa danno nelle due caselle verso cui è rivolto il protagonista, i danni sono maggiori del bastone.
- Palla Chiodata: danneggia tutti i nemici intorno al protagonista.
- Arco: spara delle frecce per attaccare a distanza. Le frecce sono infinite e causano danno all’impatto.

Nel negozio sono acquistabili dei potenziamenti, i quali restano attivi per un certo numero di livelli, dopo di che si esaurisce l’effetto. I potenziamenti disponibili sono di due tipologie.

- UltraDanno: qualunque arma abbia il protagonista permette di uccidere i nemici in un colpo solo.
- SuperScudo: il protagonista è in grado di fermare i colpi dei nemici e per questo non subisce danno.

2.3 Livelli

Ogni livello è composto da una mappa generata a partire dall'identificativo del livello. Essendo gli identificativi univoci, tutte le mappe sono diverse l'una dall'altra. All'interno della mappa sono presenti degli ostacoli e due porte ('#') che permettono di indietreggiare al livello precedente, ritrovando lo schema così come era stato lasciato, oppure avanzare al livello successivo (se possibile).

Il livello iniziale non contiene nemici per permettere all'utente di prendere confidenza con i comandi.

2.4 Nemici

Alla creazione di un nuovo livello i nemici sono generati scegliendo casualmente fra 3 possibili personalizzazioni:

- Goblin ('S'), insegue il protagonista senza però poter saltare, infligge danno solo a contatto.
- Scheletro ('{'), infligge danno sparando delle frecce che viaggiano in linea retta, il personaggio non si muove.
- Guardia ('G'), si muove orizzontalmente e causa molto danno a contatto.

I nemici sono inizialmente sempre lo stesso numero all'interno dei livelli. Il danno causabile dai nemici è scelto random tra una soglia minima e una massima, ed esso aumenta ogni 5 livelli. Il loro movimento è gestito dal pc. Il nemico muore quando la sua vita arriva a zero.

2.5 Negozio

Il negozio è sempre accessibile in qualunque momento, quando vi si entra il livello viene messo in pausa. All'interno del negozio si possono acquistare le armi, i potenziamenti e/o della vita aggiuntiva che può far sforare i 100 punti vita iniziali. Gli oggetti in vendita nel negozio sono inesauribili, quindi sempre acquistabili. All'uscita dal negozio il livello viene ripristinato come era al suo abbandono.

3 - Codice

3.1 Cartelle

- db: file per i salvataggi di stato
- src: file sorgente delle classi
    - elementi: classi di personaggi, potenziamenti...
        - armi: classi delle armi
        - personaggi: classi protagonista e nemici
        - potenziamenti: classi dei potenziamenti
    - main: classi main e principali (gestore grafica, livelli...)
    - util: classi di utilità (accesso file, stringhe)

3.2 Classi

Il protagonista ha una classe specializzata (*Protagonista*) che mantiene tutti gli attributi ed i metodi per modificarli, ad esempio per diminuire la vita o sostituire l'arma in dotazione. Implementa inoltre metodi propri come il salvataggio dello stato del personaggio sul file o la rigenerazione a seguito della morte. Nel costruttore il protagonista viene inizializzato a partire dai dati che sono salvati sul file, se la vita risulta zero vuol dire che il protagonista è da rigenerare e la partita ricomincia.

I nemici hanno tutti una classe padre (*Nemico*) che mantiene gli attributi e mette a disposizione dei metodi necessari a tutti. Le sottoclassi (*Scheletro*, *Goblin*, *Guardia*) servono per specializzare gli attributi con valori fissati.

Stessa cosa vale per le armi, che hanno una classe padre (*Arma*) la quale contiene gli attributi con i relativi getters e le sottoclassi (*Bastone*, *Spada*, *PallaChiodata*, *Arco*) specializzano gli attributi con valori fissi, diversi l'una dall'altra.

Seguono la stessa struttura anche i potenziamenti, la cui classe padre (*Potenziamento*) contiene gli attributi ed i metodi comuni, mentre le sottoclassi (*UltraDanno* e *SuperScudo*) specializzano gli attributi quali durata e costo. L'effettivo utilizzo dei poteri viene delegato al momento del bisogno nella classe del protagonista.

La classe *Negozio* gestisce lo shop in cui il protagonista può acquistare armi, potenziamenti e/o vita (inesauribili). Il negozio ha una propria interfaccia grafica che viene gestita dalla classe stessa.

Un singolo livello è gestito dalla classe *Livello*, la quale mantiene l'identificativo univoco, il flag della presenza o meno del protagonista e la lista dinamica dei nemici presenti. I metodi messi a disposizione consentono di aggiungere/rimuovere i nemici, modificare il flag e salvare il livello su file.

L'insieme dei livelli, e quindi la partita, viene gestito dalla classe *Gioco*. Questa classe contiene la lista dinamica bidirezionale dei livelli con i metodi per crearli ed aggiungerli, oltre che per muovere il protagonista tra uno e l'altro. Essa implementa inoltre i metodi per salvare e caricare i dati dell'intera partita (protagonista escluso). Al costruttore viene passato un flag che decide se resettare la partita oppure caricare i dati salvati sul file. La classe ha un puntatore al livello in cui si trova il protagonista per semplificarne l'accesso.

Tutto ciò che riguarda l'interfaccia grafica ed il movimento degli oggetti a schermo è gestito dalla classe *GUI*. Questa classe contiene i metodi per stampare la mappa con il protagonista ed i nemici e muovere i personaggi. Implementa anche i metodi che consentono l'attacco tra protagonista e nemici, oltre il metodo che gestisce l'intero gioco, attuando la gravità e sincronizzando i movimenti dei personaggi.

Due classi *Stringa* e *GestoreFile* servono per semplificare l'uso degli array di char e l'accesso ai file in lettura/scrittura. *Stringa* mantiene salvato l'array di char e dispone di metodi per concatenare, comparare e trasformare in interi altre stringhe.

GestoreFile memorizza i percorsi per i file del database ed implementa i metodi per aprire e chiudere i file, leggere e scrivere (in append o meno).

La classe *Main* contiene per l'appunto il main da cui eseguire il gioco.

La classe *Reset* contiene un main secondario per resettare il database in caso di errori inattesi.

3.3 Salvataggi

Lo stato della partita viene salvato nel file "db/partita.txt". Il file mantiene salvata la lista ordinata, dal primo all'ultimo, dei livelli (id, presenza o meno del protagonista) con i nemici presenti ed i valori dei loro attributi (i nemici solo dell'ultimo livello). L'inizio del livello è segnato con '@', i due valori successivi sono l'id seguito da 1 se era il livello attuale, 0 altrimenti. '#' segnala la fine del file.

Lo stato del protagonista viene salvato in un file a sè, ovvero "db/protagonista.txt". Il file contiene gli attributi in ordine: nome, vita, denaro, arma, potenziamento, durata potenziamento, punti, coord x, coord y, versoDestra.

4 - Scelte implementative

Alcune delle scelte principali che sono state seguite durante la realizzazione:

La lista dinamica dei livelli è stata pensata bidirezionale per semplificare lo spostamento del protagonista tra i livelli sia in avanti che indietro.

Lo schema del livello è stato suddiviso in parti della stessa estensione in modo che in fase di creazione del livello si andasse a scegliere tra dei componenti predefiniti da legare insieme per creare mappe diverse. Ogni segmento del livello contiene un solo nemico scelto casualmente tra le 3 personalizzazioni disponibili.

La classe Livello espone il type del nodo per permettere alle classi che la implementano di poter scorrere la lista dei nemici.

Le frecce sparate dai nemici di tipo Scheletro vengono memorizzate nella lista dei nemici subito dopo colui che le spara, per fare si che quando lo scheletro muore, si rimuove anche la freccia che lo segue in lista. Le frecce sparate dall’arco del protagonista vengono sempre inserite in fondo alla lista.

La classe Stringa è stata pensata per poter utilizzare più comodamente gli array di char, ad esempio per poter fare il return dai metodi oppure per concatenare gli attributi in fase di salvataggio su file (uso di overload).

La classe GestoreFile è utile per avere un solo punto di contatto all’interno del progetto tra il codice ed i file testuali, in modo da fissare i percorsi e sincronizzare le aperture e chiusure dei file.

Se i file del database perdessero la struttura pensata a causa di errori imprevisti il gioco andrebbe in crash senza possibilità di evitarlo. La classe Reset consente di sovrascrivere i dati con “quelli di fabbrica” permettendo l’avvio da zero.