Relazione di Programmazione ad Oggetti

ColiseumQT

Combatti nell'Arena

Studente : Sebastiano Zamberlan
Matricola : 1071082
Anno : 2015/2016
Sistema Operativo di Sviluppo : Ubuntu 14.04 LTS
Versione QT: Qt Creator 3.5.1(opensource) Based on Qt 5.5.1
Versione Compilatore : GCC(x86 64bit) 4.9.1

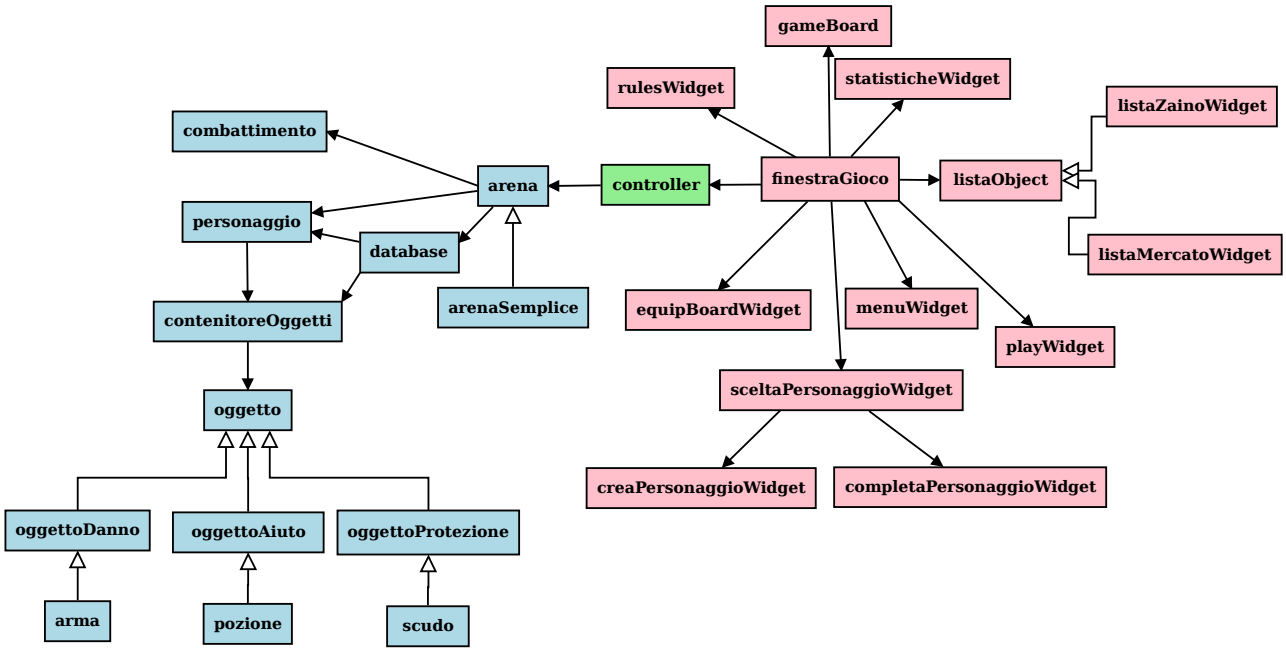
1. Scopo del Progetto

Il progetto si prefigge la creazione di un gioco stile arena rpg, in cui più personaggi si sfidano a turno all'interno di un'arena. Tramite mosse e abilita si può combattere ed eseguire attacchi e difendersi da questi attacchi. Ogni personaggio ha un equipaggiamento in cui può avere armi, armature, scudi e pozioni con cui aumentare il danno, proteggersi da altri danni, parare dei colpi e rigenerare la salute e l'energia.
Un attacco può andare a buon fine o meno lo stesso vale per le parate.
Il progetto quindi si prefigge la creazione di quanto più simile a tutto questo.

Una volta creato il tuo personaggio attraverso una serie di possibilità* il gioco prevede una serie di avversari da superare per vincere l'arena. Oltre a questo il gioco prevede uno zaino in cui il proprio personaggio può depositare gli oggetti per poi poterli vendere al mercante ove è anche possibile comprare oggetti. Ci sarà uno spazio dedicato all'equipaggiamento del giocatore dove si potrà decidere cosa il proprio giocatore dovrà indossare o utilizzare.
Ogni vittoria all'area comporta una vincita in soldi che permette di comprare poi nuove armi più forti e più costose dal mercante.
Il gioco è stato pensato per essere senza salvataggi in quanto l'abilità sta nel battere nel minor tempo possibile tutti i nemici. Le informazioni sui nemici e il mercato vengono però raccolte da un database.

Il progetto è stato creato quanto più possibile espandibile ed estendibile con la presenza di 2 gerarchie di classi e di una classe che ha ampi margini di estensibilità. Si utilizza molto codice polimorfo che semplifica di molto la gestione dei contenitori di oggetti.
La gerarchia di classe principale è oggetto che rappresenta gli oggetti utilizzati, poi c'è la classe personaggio che rappresenta i personaggi che si dovranno sfidare e arena che rappresenta l'arena in cui combattere. Da notare la presenza di un contenitore di oggetti e una classe chiamata combattimento completamente adibita al combattimento fra 2 personaggi.
C'è un unica classe controller che occupa delle operazioni logiche necessarie alle varie classi grafiche.
Sono presenti poi 13 classi grafiche che si occupano di gestire le varie viste dell'applicazione.

* Queste possibilità sono solo di interfaccia in quanto per rendere il gioco più equilibrato ho deciso di far scegliere al giocatore tra 3 razze e poi tra 2 classi , la scelta di una o l'altra identifica una diversa serie di statistiche scelte da me per poter godere di un gioco quanto meno equilibrato e con una certa sfida. Le caratteristiche di un personaggio verranno poi visualizzate nelle statistiche.



2. Descrizione della Gerarchia Oggetto

2.1 Oggetto

Con questa classe si intende definire un oggetto che possa essere utilizzato da una classe personaggio per vari scopi come arrecare danno, protezione ecc. Ogni oggetto possiede 4 campi che si riferiscono al nome dell'oggetto, al peso dell'oggetto alla sua grandezza e al suo costo.
Ci sono vari metodi virtuali che identificano quello che un oggetto può fare in generale ma che in questa classe non è ancora implementato in quanto non si conoscono le informazioni per tale conoscenza ma le classi che ne deriveranno implementeranno tali metodi. E' una classe base astratta.

2.1.1 OggettoDanno

Deriva da Oggetto e ne implementa i metodi virtuali, in quanto un oggettoDanno è definito come un oggetto in grado .di creare del danno (o in generale di offendere) e di creare un valore che corrisponde ad uno sforzo per generare il danno . Inoltre ci sarà una probabilità data dall'arma che identifica la probabilità di eseguire il danno. Tutti gli altri metodi implementati ritorneranno 0 in quanto un oggettoDanno almeno per come è definito non ha la possibilità di proteggere o di generare un aiuto.
Ci sono 3 campi dati nuovi che sono danno, probabilità di danno e manovrabilità che corrispondo alle caratteristiche definite precedentemente.

2.1.1.1 Arma

Deriva da OggettoDanno ed esegue override su alcuni metodi, esso corrisponde ad un oggetto in grado di fare varie tipologie di danno a seconda del come lo si utilizza e che in base alla mossa eseguita generi un valore diverso sia di danno che di sforzo necessario a compierlo, le differenze sostanziali nelle mosse sono dovute alla probabilità che un arma ha di colpire. Inoltre anche le mosse hanno una loro probabilità di efficacia che corrisponde alla percentuale di superficie che arreca danno . Ci sono 2 nuovi campi dati che sono il danno critico dell'arma e la percentuale di superficie che arreca danno, inoltre in questa classe la manovrabilità viene impostata a 1.0 in quanto un arma di base non ha difficoltà di manovrabilità (nell'eseguire un semplice movimento per arrecare danno) ma successivamente all'utilizzo di una mossa il valore muterà e con ciò muterà anche lo sforzo ottenuto nel manovrare l'arma.

2.1.2 OggettoProtezione
Deriva da Oggetto e ne implementa i metodi virtuali, in quanto un oggettoProtezione è definito come un oggetto in grado di fornire una protezione e di creare un valore che corrisponde ad un deficit nell'utilizzo di questa tipologia di oggetti. C'è un nuovo campo dati che corrisponde all'effettiva protezione fornita.

2.1.2.1 Scudo
Deriva da OggettoProtezione ed esegue l'override su alcuni metodi virtuali. Questa classe rappresenta un oggetto che oltre a fornire una protezione fornisca anche una probabilità di parata. La differenza fra parare e proteggersi sta nel fatto che parare permette di non subire il colpo mentre proteggersi non lo implica. Di conseguenza uno scudo può fornire da protezione ma può anche fornire una probabilità di parata che corrisponde alla possibilità di fermare una danno che si riceve. C'è un nuovo campo dati che rappresenta la probabilità di parare dello scudo.

2.1.3 OggettoAiuto
Deriva da Oggetto e ne implementa i metodi virtuali, un oggettoAiuto è definto come un oggetto in grado di fornire un valore che valga da aiuto per qualsiasi necessità. C'è un nuovo campo dati che corrisponde all'effettivo valore dell'aiuto.

2.1.3.1 Pozione
Deriva da OggettoAiuto e ne overrida alcuni metodi virtuali. Una pozione è definita come un oggetto in grado di fornire uno specifico aiuto in base alla tipologia di richiesta e in base alla tipologia di aiuto in grado di fornire. Essa sarà in grado di fornire un aiuto solo se la richiesta sarà uguale a quella della tipologia presente in pozione. Inoltre l'oggetto pozione ha n(inteso come numero da definire nella creazione) utilizzi e quando scenderanno a 0 cesserà di essere utile e il suo prezzo come ovvio scenderà a 0. Ci sono 2 nuovi campi dati che rappresentano la tipologia e il numero di utilizzi.

2.2 ContenitoreOggetti
Rappresenta un contenitore di oggetti polimorfi.

Quindi per definire l'utilità dei tipi oggetto essi rappresentano oggetti che possono essere utilizzati per vari scopi e che a seconda del loro tipo provocheranno effetti diversi nelle invocazioni dei metodi virtuali.

3. Model e Descrizione del Polimorfismo

3.1 Personaggio
Con questa classe si intende definire un personaggio in grado di eseguire danni e di proteggersi tramite o senza l'utilizzo di oggetti. Possiede un certa probabilità di attacco e una certa probabilità di Difesa, essi rappresentano la possibilità di eseguire un danno o di evadere (inteso come schivare) da questo. Esso è in grado di utilizzare la classe polimorfa oggetto. Esso possiede due contenitoriOggetti che rappresentano lo zaino del personaggio e il suo equipaggiamento. Tramite polimorfismo si gestisce l'equipaggiamento che sarà rigorosamente formato da un oggetto in grado di creare danno , un oggetto in grado di proteggere e che sarà utilizzato solo per questo scopo. Un oggetto in grado di parare e di proteggere e 2 oggetti in grado di fornire un aiuto.

Il codice polimorfo si utilizza per l'inserimento degli oggetti in equipaggiamento in quanto solo se un oggetto è in grado di fare danno potrà essere inserito nella “slot” dell'arma e così via per gli altri. In particolare ci sarà un test se il danno è > 0 e solo in questo caso l'arma verrà inserita.
Stesso procedimento vale per gli altri casi, questo rappresenta un test sulle classi virtuali di oggetto e che quindi genereranno il polimorfismo. Lo stesso dicasi per i metodi che eseguono uno scambio tra oggetti dello zaino e oggetti nell'equipaggiamento.
I metodi virtuali di personaggio, in particolare attacca() e difendi() sono ancora un esempio di polimorfismo per quanto riguarda la gerarchia oggetto.

3.2 Arena
Arena rappresenta, come sottolinea il nome, un arena di lotta fra 1 personaggio e un vector di personaggi avversari con anche un mercato in cui comprare gli oggetti. Essi combattono tramite un classe combattimento che ne definisce la vittoria o la sconfitta. Questa classe possiede come campi dati un puntatore ad un database contenente un vector di puntatori a personaggi e un contenitore di oggetti che fungerà da mercato, inoltre arena ha una puntatore a una classe combattimento, un puntatore a personaggio che rappresenta colui che dovrà affrontare i restanti avversari e un numero che rappresenterà il n° di avversari sconfitti e quindi il livello a cui si è giunti. Arena è una classe base astratta.

3.2.1 Arena Semplice
Deriva da arena e corrisponde al modello facile di arena in quanto implementa direttamente un nuovo database. Essa implementa la gestione del combattimento fornendo una sorta di rudimentale intelligenza artificiale per gli avversari.
Inoltre implementa il metodo che rappresenta la vittoria e quindi gestisce e decide il metodo di vittoria che in questo caso sarà banalmente la sconfitta di tutti gli avversari.

3.3 Database
Classe con un vector di personaggi e un contenitore di oggetti.
Oltre ai contenitori sono presenti i metodi necessari per l’inserimento e la rimozione.

3.4 Combattimento
La classe combattimento rappresenta una lotta fra 2 personaggi. È implementata tramite 2 puntatori a personaggio. Essa ha contiene vari metodi che definiscono il combattimento e la vittoria o la sconfitta di ciascuno dei due personaggi.

4. Controller

4.1 Controller
La classe controller è l’unica classe di tipo controller presente nel progetto. Essa gestisce la maggior parte delle operazioni logiche necessarie al corretto funzionamento delle classi grafiche. Viene creata mediante l’apposito costruttore all’interno della classe finestraGioco, dove viene fornita alle altre classi grafiche mediante un opportuno puntatore.
Essa ha come campi dati oltre a quelli utili alla parte grafica anche un puntatore alla classe arena su cui si utilizzerà il polimorfismo per la gestione principalmente del combattimento e ricavare tutte le informazioni utili alla classe grafica.

5. View

In seguito alle correzioni segnalate durante la scorsa consegna le classi grafiche del progetto hanno subito estese modifiche. In particolare precedentemente il progetto consisteva in un'unica classe grafica che tramite i suoi metodi generava viste diverse dell'applicazione. Sebbene tuttavia l'idea sulla quale si sono sviluppate le classi rimane simile precedentemente c'era un notevole spreco di codice e di ripetizione in quanto venivano creati numerosi metodi solo per spostarsi da una view all'altra. Ora invece si sono create ben 9 classi diverse che rappresentano una vista diversa dell'applicazione. Non ho potuto inserire una grande gerarchia in quanto le classi differenziavano molto tra di loro sia a livello di set del layout e anche di widget che anche se una minima parte(2-3) risultavano in comune avevano funzioni diverse nelle varie classi in cui venivano creati. Si riscontra comunque la presenza di una gerarchia che è quella di listaObject.
Ogni classe è implementata graficamente in maniera molto diversa dalle altre di conseguenza le varie classi sono state create separatamente, tranne listaObject.
In seguito alla creazione di queste classi si è risparmiata una notevole quantità di codice.
Brevemente qui sotto definisco i principali scopi e utilità delle varie classi grafiche.
Tutte le view ottengono i dati e le informazioni tramite il controller.

5.1 FinestraGioco

Rappresenta la classe che contiene tutte gli altri widget e che tramite il controller modifica la vista a seconda delle scelte.

5.2 PlayWidget

Gestisce la prima vista del gioco, in particolare la semplice scelta se giocare o meno.

5.3 SceltaPersonaggioWidget

Rappresenta la vista che gestisce la creazione di un personaggio, essa è composta da 2 widget (creazionePersonaggio e completaPersonaggioWidget) che rappresentano i 2 step della creazione.

5.4 MenuWidget

Rappresenta la vista che gestisce la pagina principale durante lo svolgimento del gioco essa è formata principalmente da 6 pulsanti.

5.5 GameBoard

Gestisce la fase di combattimento è rende visibile tramite controller la vittoria o la sconfitta del proprio personaggio e l'efficacia dei danni apportati o subiti.

5.6 ListaObject

Classe base astratta che rende visibile una contenitore per una lista di pulsanti. Da lei derivano ListaZainoWIdget e ListaMercatoWidget.

5.7 ListaZainoWidget

Deriva da listaobject, inserisce all'interno del contenitore di pulsanti un insieme di pulsanti che rappresentano l'inseme degli oggetti presenti nello Zaino.

5.8 ListaMercatoWidget

Deriva da listaobject, inserisce all'interno del contenitore di pulsanti un insieme di pulsanti che rappresentano l'inseme degli oggetti presenti nello Mercato.

5.9 EquipBoardWidget
La classe gestisce a livello grafico gli oggetti presenti nell'equipaggiamento, ovviamente tramite controller.

5.10 RulesWidget
Mostra graficamente l'inseme di regole del gioco essendo un inseme molto lungo è stato introdotto uno scroll widget che permette di visualizzarlo meglio

5.11 Statistiche Widget
Mostra graficamente le statische del personaggio

6. Guida alla GUI

Aperta l'interfaccia grafica si otterrà subito la visione del titolo del progetto e due pulsanti che permettono uno di continuare e di iniziare il gioco, GIOCA appunto e uno di uscire immediatamente ESCI
Premuto il pulsante GIOCA si inizierà la creazione del personaggio, si potrà scegliere la razza tra tre scelte : orco, elfo e umano e inoltre in alto c'è la possibilità di inserire il nome del personaggio tramite i bottoni si potrà scegliere la razza.
A seconda della razza e premuto il bottone continua si potranno scegliere 2 diverse classi e di seguito una delle 7 abilità possibili che definiscono 7 diversi bonus.
Una volta selezionata l'abilità si potrà continuare con la creazione definitiva del personaggio tramite bottone apposito.
Conclusasi la creazione si visualizzerà il menù principale formato da 6 bottoni e una label che indica il numero di nemici sconfitti al momento e il numero di avversari ancora da battere.
I 6 bottoni portano a 6 differenti scenari.
Il bottone Combattimento in arena permetterà di visualizzare la zona di combattimento e di combattere appunto contro l'avversario attuale.
Ci saranno 4 pulsanti che rappresenteranno le 4 mosse possibili, altri 2 che permetteranno di utilizzare le pozioni se equipaggiate. Ci sarà inoltre un pulsante che permetterà di visualizzare le informazioni relative alle mosse. Per il resto sono presenti delle ProgressBar che rappresenteranno la vita attuale e la difesa attuale dei personaggi, sia del giocatore che dell'avversario.
Inoltre sono presenti anche le informazioni sull'avversario. Premendo su una qualsiasi dei pulsanti a parte quello delle informazioni, si partirà con il turno e l'avversario comincerà anche lui ad attaccare. Perde chi raggiunge per primo lo 0% in vita o
energia.
Il bottone Mercante permetterà di visualizzare gli oggetti del mercante, i quali se selezionati mostreranno le loro informazioni, in basso a destra è presente il totale dei soldi che hai a disposizione. Si possono comprare gli oggetti che verranno inviati al tuo zaino.
Il bottone Zaino permette di visualizzare gli oggetti presenti nello zaino e di,eventualmente selezionati, venderli al mercato o equipaggiarli al giocatore.
Il bottone Equipaggiamento permette di visualizzare gli oggetti presenti nell'equipaggiamento e ed eventualmente di spostarli nello zaino.
Il bottone Statistiche visualizza le statistiche del giocatore.
Il bottone Regole visualizzerà semplicemente le regole del gioco e quindi consiglio vivamente di vederlo nel caso questa guida non sia stata del tutto soddisfacente.